Titre: Title:	Control systems for experiments in quantum communication and computing on optical fibres
Auteur: Author:	David Alfonso Guzman
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Guzman, D. A. (2008). Control systems for experiments in quantum communication and computing on optical fibres [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8217/

UNIVERSITÉ DE MONTRÉAL

CONTROL SYSTEMS FOR EXPERIMENTS IN QUANTUM COMMUNICATION AND COMPUTING ON OPTICAL FIBRES

DAVID ALFONSO GUZMÁN
DÉPARTEMENT DE GÉNIE PHYSIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE PHYSIQUE)
DÉCEMBRE 2008

Your file Votre référence
ISBN: 978-0-494-48920-8
Our file Notre référence
ISBN: 978-0-494-48920-8

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONTROL SYSTEMS FOR EXPERIMENTS IN QUANTUM COMMUNICATION AND COMPUTING ON OPTICAL FIBRES

présenté par: GUZMÁN David Alfonso

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. FRANCOEUR Sébastien, Ph.D., président

M. GODBOUT Nicolas, Ph.D., membre et directeur de recherche

M. DAVID Jean-Pierre, Ph.D., membre et codirecteur de recherche

M. AZAÑA José, Ph.D., membre

ACKNOWLEDGMENTS

I would like to thank my research director, Nicolas Godbout at the Physics engineering department (génie physique) for letting me work on the quantum experiments of the Fibre optics Laboratory in a way that I put both physics and electronics knowledge in practice; for his interest in experimental applications of quantum communication and computing; for his financial support.

At the same time, I would like to thank my research co-director, Jean-Pierre David at the Electric engineering department (génie électrique) for his great support and dedication; for opening his door and mind to other domains in science; for making this project possible.

For allowing me to get involved in their experiments, I thank also Yasaman Soudagar and Guido Berlín, for sharing their knowledge and for several discussions. Their experience in the domain was really valuable to me.

To Mikaël Leduc for his technical experience, his help and advice. To François Busque, Stéphane Virally and Suzanne Lacroix who permittes me to work on their laser stability measurement project. Also to Stéphane Virally for both professional and personal talks.

To everyone who received me in this land. For their lessons, patience and interest. Among them Jean-Simon Corbeil deserves here a special mention.

To my parents for their trust and support, unconditional as usual.

RÉSUMÉ

Un système électronique utilisable pour une certaine catégorie d'expériences en information quantique est réalisé. Les spécifications du système lui permettent de répondre aux besoins particuliers de certaines protocoles (expériences de pile ou face et génération d'états « *cluster* »).

Le système réalisé au cours de ce projet est flexible et permet une adaptation à d'autres expériences par simple reprogrammation. Il permet en particulier d'accomplir des tâches utiles à la détection de photons uniques, aux comptes de coïncidences, à la détection d'événements, à la génération de nombres aléatoires, et au contrôle de phase et d'amplitude de signaux optiques.

Le cœur du système est constitué d'un circuit logique programmable (*field-programmable gate array* en anglais), qui communique avec un ordinateur via USB. Les ports externes d'entrée-sortie servent à faire la liaison entre l'unité de traitement et les périphériques. Les signaux de sortie de détecteurs de photons et d'autres périphériques sont gérés par un circuit imprimé dont l'architecture a été dessinée au cours du projet, et qui a été réalisé sur commande. Un second circuit imprimé, permettant la gestion de modulateurs électro-optiques, a également été réalisé.

Le système peut être géré par l'intermédiaire d'interfaces graphiques simples ; certaines ont été programmées à titre d'exemple. La modification du code du système se fait par l'intermédiaire de fichiers `dll` standards, dont la programmation en langage C est relativement aisée.

ABSTRACT

An electronic system enabling certain classes of quantum information processing experiments is designed and assembled. Requirements for the development of this system were obtained by studying particular quantum optical experiments, such as the implementation of coin tossing protocols, and cluster state generation.

This project provides a flexible, reprogrammable electronic system. It addresses required utilities such as single and coincident photon detection, time stamping of events, random number generation, and optical phase and amplitude control.

The processing takes part in a field-programmable gate array, which has a USB communication channel for connection to any computer. External input/output ports serve to link the processing unit and peripheral devices. Outputs from different kinds of photon detectors and similar devices are processed by a custom designed printed circuit. Electro-optic modulators require signals that change their voltage levels during a single experience; a second printed circuit was designed to deal with this situation.

Graphical user interfaces are possible with the current system; some of them were done. Generation of `dll` libraries that contains C code instructions suffices to program the card, send and receive both instructions and data during an execution of the loaded program.

CONDENSÉ EN FRANÇAIS

Introduction

La technologie en informatique quantique se développe en permettant aujourd'hui d'avoir des implémentations physiques réalisables. Elle promet des systèmes de cryptographie sécuritaires et des solutions à des problèmes computationnels que les ordinateurs classiques ne sont pas capables de résoudre, ou qu'ils peuvent résoudre de façon moins efficace. Grâce à ces promesses le sujet de recherche est devenu intéressant et actif.

Les systèmes électroniques sont une partie incontournable de ce type d'expériences ; ils permettent l'enregistrement d'un grand nombre de mesures et le contrôle d'appareils.

Typiquement, on affronte les besoins en électronique en utilisant des appareils de type « boîte noire », capables de remplir une seule fonction spécifique. En plus, il est possible que ce type d'appareils fonctionne avec un signal d'entrée très spécifique et pas nécessairement conventionnel. L'alternative est de créer des systèmes électroniques au besoin.

L'intérêt de concevoir des systèmes d'information quantique et les défis que les systèmes électroniques posent à leur fabrication forment la principale motivation de ce projet. L'absence de systèmes commerciaux qui fonctionnent avec différent sortes de détecteurs et leurs signaux, et l'incompatibilité entre les étages d'acquisition, traitement et d'exécution des expériences ont encouragé la réalisation de ce projet.

L'objectif de cette recherche est donc de réaliser un système électronique pour contrôler les appareils de mesure et de contrôle utilisés en expériences d'informatique quantique basées sur fibres optiques. Les implémentations optiques sont au centre du projet. On désire une solution flexible qui marche dans le plus grand nombre de scénarios.

Parmi les exigences du système, on doit acquérir et enregistrer les données provenant des détecteurs de photons. La détection de coïncidences entre détecteurs est requise pour observer et exploiter l'intrication, propriété donnée par la mécanique quantique. Par ailleurs, le contrôle de modulateurs électro-optiques de phase et d'amplitude est important pour générer l'information encodée optiquement. Il nous permet d'utiliser la superposition, une deuxième propriété offerte par la mécanique quantique.

La solution proposée remplit ces exigences. Il est conçu sur une carte électronique re-programmable, un FPGA (« *field-programmable gate array* »), qui peut communiquer avec un ordinateur pendant l'exécution d'une routine en temps réel. La standardisation des signaux d'entrée est faite par un circuit imprimé fait sur mesure. L'amplification de quelques signaux de sortie est requise, et est réalisée par un deuxième circuit imprimé fait sur mesure.

Des tests des applications programmées sont présentés et analysés. Ils démontrent l'utilité, la flexibilité et la performance du produit ici développé.

Contenu

Dans ce projet, l'information quantique est encodée sur des photons. Leur information peut être encodée sur des états de polarisation, où la base habituelle est $|\phi\rangle = \alpha|H\rangle + \beta|V\rangle$ (horizontal et vertical). Également, elle peut être encodée sur des états d'encodage temporel (*time-bin encoding*), avec la base $|\phi\rangle = \alpha|s\rangle + \beta|l\rangle$ qui correspond aux chemins court $|s\rangle$ (*short* en anglais) et long $|l\rangle$ d'un interféromètre de Mach-Zender.

L'information temporelle de la génération des photons (signal de synchronisation du laser), et leur détection (détecteurs de photons individuels) peut être utilisée pour synchroniser l'expérience, processus utile pour déterminer la corrélation entre les photons

émis et mesurés. Ceci impose la réception de différents types de signaux électriques, et leur traitement sur un pied d'égalité. Un module de conversion analogique-digital est réalisé.

Pour déterminer l'état quantique, on fait des mesures sur différentes bases. La détection des coïncidences et anti-coïncidences est requise pour faire cette discrimination.

La démonstration des protocoles de pile ou face quantique ou de cryptographie quantique nécessitent la génération de nombres aléatoires classiques. Il est possible de les générer à partir d'un système électronique produisant des séquences de bits pseudo-aléatoires.

L'utilisation d'un interféromètre pour l'encodage et le décodage temporel des photons nécessite le contrôle d'un modulateur de phase. Ce modulateur accepte un signal électrique périodique, laquelle détermine avec son amplitude la phase qui sera appliqué sur le signal optique.

L'implémentation des protocoles quantiques nécessite l'acquisition de différents types de signaux électriques. Le système d'acquisition développé dans cet objectif permet d'acquérir aussi bien des signaux TTL de 1,8 V d'une durée d'environ 150 ns que des impulsions irrégulières autour de 100 mV d'amplitude et de durées de l'ordre de 3 ns. Ces signaux sont typiques des sorties de détecteurs de photon (Si ou InGaAs) et des impulsions de synchronisation de laser pulsés.

Le composant choisi pour réaliser la conversion des signaux est un Micrel SY58601U (*ultra-precision differential* 800 mV *LVPECL line driver/receiver with internal termination*). Il permet de générer un signal différentiel qui peut être lu par un FPGA. Un PCB (circuit imprimé) capable de réaliser la conversion de 8 canaux a été développé. Les signaux produits par le PCB sont ensuite acheminés vers la carte FPGA qui les traite.

Le FPGA utilisé est un Spartan-3 de Xilinx, intégré à une carte électronique ZestSC2

d'Orange Tree Technologies. En plus du FPGA, la carte intègre un système de communications USB, une mémoire SDRAM, une mémoire flash, et des diodes électroluminescentes (DELs). La carte est programmable en C, et des exemples de configuration en VHDL et verilog, un fichier ucf pour configurer le FPGA et quelques applications exécutables, sont fournis.

Plusieurs applications ont été réalisées pour répondre aux besoins particuliers des expériences d'information quantique. En particulier, la mesure de compte d'évènements, la détection de coïncidences, et l'enregistrement d'information temporelle ajoutée aux données issues de l'acquisition ont été développées au cours du projet.

La génération de numéros aléatoires et la génération de signaux périodiques sont permettent d'offrir un signal de sortie, qui peut gérer des modulateurs électro-optiques. Le FPGA peut générer la porteuse pour ces signaux. Mais il est nécessaire de moduler l'amplitude du signal pour gérer les modulateurs également.

Un deuxième circuit imprimé est fabriqué pour amplifier des signaux périodiques. Une modulation de phase quelconque ϕ s'obtient avec un voltage appliqué sur le modulateur de phase de $V_\phi = \phi V_\pi/\pi$, où V_π détermine le voltage nécessaire pour avoir une phase optique de π ; une valeur typique est $V_\pi = 5$ V.

Les amplificateurs opérationnels ne possèdent pas la bande-passante adéquate pour amplifier les signaux aux fréquences désirées. Pour le cas où il faut travailler à quelques centaines de MHz, l'utilisation de transistors comme interrupteurs est nécessaire. Le transistor choisi est le PD57002-E (*RF power transistor* de STMicroelectronics). Il est conçu pour fonctionner à haute fréquence jusqu'à 1 GHz.

Les protocoles de communication quantique requérant un encodage temporel (*time-bin*), utilisent des valeurs discrètes et prédéterminées de modulation de phase. Les valeurs typiques pour ce genre d'application sont des multiples de $\pi/4$, mais d'autres valeurs

peuvent être requises pour des applications particulières.

La solution implémentée pour obtenir plusieurs valeurs de modulation consiste à des branches partagées par un connecteur. Chaque branche possède une résistance différente et un transistor qui gère le courant dans la branche. Les sorties logiques (0 ou 1) du FPGA déterminent quelles branches s'activent, ce qui permet de moduler la tension de sortie.

Le circuit imprimé possède 4 sorties : deux d'entre elles génèrent 6 niveaux de tension différent ; les deux autres ne délivrent qu'un seul niveau de tension Les modulateurs de phase sont gérés par les deux premières et les modulateurs d'amplitude par les deux autres. Le même circuit peut être reproduit sur deux ports de la carte ZestSC2, ce qui permet de gérer jusqu'à huit appareils simultanément : quatre modulateurs de phase et quatre modulateurs d'amplitude.

Conclusion

Un système de traitement de données d'expériences en communication et calcul quantique a été réalisé. Le système développé offre une solution flexible et adaptable à de nombreuses situations grâce au circuit imprimé conçu, réalisé et testé au cours du projet.

Le circuit d'entrée gérant la standardisation des signaux a été testé dans des conditions variant en amplitude de 100 mV à 1 V et en cycle d'utilisation de 4 à 96%. La sortie du circuit fournit un signal différentiel de ± 800 mV dont la fréquence suit celle du signal d'entrée.

Des applications particulières ont été réalisées, comme la mesure de compte d'évènements, la détection de coïncidences, l'enregistrement d'information temporelle, la génération de nombres pseudo-aléatoires, et la génération de signaux périodiques. La routine

de compte d'évènements performe de manière exceptionnelle, avec un taux d'erreur sur la fréquence mesurée ne dépassant pas 0,002 5%.

La détection de coïncidences a été réalisée de deux manières : synchrone et asynchrone. Les deux méthodes sont capables de correctement distinguer entre des scenarios avec et sans coïncidences. Pour la version asynchrone, la fenêtre de coïncidence a une durée de 1,35 ns. Une version synchrone équivalente aurait besoin d'une horloge nterne proche de 4 GHz. Pour la version synchrone, la fenêtre de coïncidence dépends de l'horloge du système ; avec $T_{\text{CLK}} = 200$ MHz la durée d'ouverture de la fenêtre et comprise entre 10 ns et 15 ns.

L'application d'enregistrement d'information temporelle permet de reconstruire jusqu'à 5 signaux numériques. Le nombre de signaux enregistrés peut être modifié, ce qui entraîne un changement dans le temps de débordement. L'information temporelle ajoutée est requise pour la détection de coïncidences, l'étude de la corrélation entre les signaux et les statistiques d'arrivées. Cela est fondamental pour les expériences de communication quantique, où le taux de comptes est entre 100 et 10 000 comptes par seconde.

Pour contrôler des modulateurs de phase et d'amplitude, des signaux périodiques ont été générées à 45 MHz et 96 MHz. Leur amplitude est amplifiée (ou réduite) en utilisant le circuit imprimé conçu et fabriqué dans ce but. Ce circuit a été testé dans des conditions prouvant que ses caractéristiques sont suffisantes pour obtenir la modulation de phase et d'amplitude requises. Pour un fonctionnement nominal entre 0 V et 5 V, les tensions de sortie étaient de 0,63 V et 4,67 V respectivement. La réponse en fréquence a été également testée avec succès.

Une séquence de nombres pseudo-aléatoires a été générée à l'aide d'un LFSR Galois. Une telle séquence peut être utilisée dans un générateur de pile ou face et dans les protocoles de communication quantique nécessitant d'une sélection aléatoire d'éventements.

Le système développé présente plusieurs avantages. En particulier, sa flexibilité, sa capacité de reprogrammation, son adaptation facile à d'autres sujets permettent une expansion facile du projet. En plus, il offre des interfaces graphiques faciles à gérer, comme celles qu'ont été réalisées avec LabView.

Le codage utilisé pour programmer le FPGA est spécifique à la carte électronique Zest-SC2. Mais grâce au design par bloc, il est possible de l'adapter pour d'autres cartes électroniques, y compris celles provenant de manufacturiers différents.

La principale barrière technologique de ce travail est la vitesse de traitement. L'horloge du système fonctionne normalement à 48 MHz. L'utilisation d'autres fréquences peut impliquer des modifications importantes. De plus, la plage des fréquences pouvant être générées par le FPGA est limitée, et la carte électronique utilisée dans ce projet ne supporte pas de fréquences plus élevées que 210 MHz.

Quand on l'utilise avec des systèmes de fibres optiques utilisant un encodage temporel (*time-bin encoding*), la vitesse maximale devient une limitation de taille pour l'interféromètre optique. Par exemple une horloge à 200 MHz (5 ns) provoque un retard de 1,5 m dans l'interféromètre.

Il est recommandable de prolonger ce projet. Des nouvelles applications et fonctions peuvent être ajoutées et intégrées, de telle façon que le produit final soit plus avantageux. Le produit pourrait être utilisé pour différents sujets de recherche.

Un chercheur qui voudrait continuer avec ce projet devrait partir des applications ici déjà développées, les exécuter et comprendre leur codage dans les différents langages de programmation (VHDL, C, LabView). Comme premier tâche, je suggère d'améliorer la détection de coïncidences en réglant les temps de délai internes du FPGA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF NOTATIONS AND SYMBOLS

$\oplus$: Addition modulo 2

$\hat{\oplus}$: *Controlled-not* quantum operator

$\otimes$: Tensor product

$||$: Parallel to (resistances or capacitors)

$\mathbb{C}$: Complex numbers

$\mathbb{N}$: Natural numbers

$\hat{U}$: Quantum operator

$\hat{U}^{\dagger}$: Hermitian conjugate of a quantum operator

η : Quantum efficiency

λ : Wavelength

f : Frequency

I : Current

n : Refractive index

P : Power

Q : Positive signal of differential pair output

$/Q$: Negative signal of differential pair output

R : Resistance

t : Time

T : Period

V : Voltage

x : Distance

Z : Impedance

Acronyms and abbreviations

AC : Alternating current

APD : Avalanche Photodiode

BNC :	Bayonet Neill Concelman
CLB :	Configurable Logic Block
CLK :	Clock
DC :	Direct current
DCM :	Digital Clock Manager
DLL :	Dynamic Link Library
FIFO :	First In, First Out
FPGA :	Field-programmable gate array
GND :	Ground
GUI :	Graphical User Interface
I/O :	Input/output
IOB :	Input/Output Block
LED :	Light-emitting-diode
LFSR :	Linear feedback shift register
LSB :	Least significant bit
LUT :	Look-Up Table
LVCMOS :	Low Voltage Complementary Metal Oxide Semiconductor
LVPECL :	Low-voltage positive emitter-coupled logic
LVTTL :	Low Voltage Transistor-transistor logic
MLF :	MicroLeadFrame
MSB :	Most significant bit
PCB :	Printed circuit board
PBS :	Polarization beam splitter
PRNG :	Pseudo-random number generator
RAM :	Random Access Memory
RF :	Radio frequency

RNG :	Random number generator
RTL :	Register Transfer Level
SDRAM :	Synchronous dynamic random access memory
SMA :	SubMiniature version A
SMD :	Surface-mount device
TTL :	Transistor-transistor logic
UCF :	User constraints file
USB :	Universal Serial Bus
VHDL :	VHSIC hardware description language
VHSIC :	Very-High-Speed Integrated Circuits
XST :	Xilinx Synthesis Technology

LIST OF TABLES

LIST OF APPENDICES

INTRODUCTION

Quantum information theory and technology have developed in such a way that, nowadays, its practical implementation is starting to arise. It promises unconditionally secure cryptographic systems and solutions to computational problems that current classical computers solve either inefficiently of not at all, reasons that make it an attractive and active research subject.

Electronic systems are unavoidably involved in this process; they are tied to measurement devices and external controllers, regardless of the experiments' quantum nature. The most common existing approach is to deal with electronic systems using specific function black-boxes, where each of them serves a single purpose; most of the time they may only work with a determined non-conventional type of signal. A second approach is to make custom solutions when required.

This project born moved by the interest in making quantum information systems feasible, and due to the diversity of challenges that electronic systems offer to such implementations. The lack of commercial systems that work for different type of detector signals, and the disconnection between acquisition, processing and action stages are among the reasons that encouraged the realization of this project.

Hence, the purpose of the present investigation is to develop an electronic system to control the action and measurements devices used in quantum information experiments based on optical fibres. Optical schemes being the focus of this project, are specifically targeted. A flexible solution, which works in as many contexts as possible, is pursued.

Electronic requirements for running quantum information experiments on optical schemes are identified and addressed. They require acquisition and recording of data coming from photon detectors. Recognition of coincidences between detectors is required to detect

and exploit a quantum mechanical property named entanglement. Controlling phase and amplitude electro-optical modulators is relevant in order to manipulate the information on optical systems; this enables harnessing superposition, a second property provided by quantum mechanics. For experiments with more than one party, providing a classical communication channel is necessary. Finally, it is important to synchronize the whole system.

The proposed solution takes into account these requirements. It is based on a reprogrammable electronic device, a field-programmable gate array, which can communicate with a computer while executing a routine in real time. Standardization of incoming signals is done by a custom designed printed circuit. Amplification of some outgoing signals is required, and is done by a second custom designed printed circuit.

Tests of programmed applications are presented and analysed. They demonstrate the utility, flexibility and performance of the product here developed.

CHAPTER 1

BASICS IN QUANTUM COMMUNICATION AND COMPUTING

To operate experimentally either a quantum communication or a quantum computing system, many electronic tasks are required. The purpose of this project is to provide an electronic system as general as possible for these situations. The quantum information domain is treated before entering the electronics domain, to illustrate the requirements and relevance of the developed system.

Quantum information offers characteristics that are not available in its classical counterpart, and makes possible solving some problems for which current algorithms are either innefficient or inexistant. Superposition and entanglement properties allow quantum schemes be differentiated from classical ones. The theory behind this statement is presented in this chapter, along with the experimental optical implementation schemes that enclose this project.

Of particular interest is the one way quantum computing scheme proposed by Raussendorf and Briegel [Raussendorf and Briegel, 2001] as well as quantum communication problems, like the quantum key distribution intended for cryptography purposes and the quantum coin tossing protocol [Bennett and Brassard, 1984].

1.1. Quantum information

This scheme of computing is based on the quantum mechanical properties provided by its minimal information unit, known as a qubit (quantum bit). They can be obtained by any quantum system of two levels, which are represented as $|0\rangle$ and $|1\rangle$.

While a classic bit can only take one of the values, 0 or 1, a quantum state is able to be in a *superposition* of both basic states $|0\rangle$ and $|1\rangle$. This means that in a quantum system it is possible to be in both values at the same time, each of them with a certain probability. The sum of these probabilities must be as usual 1. In other words

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

where $|\psi\rangle$ represent the qubit state, and $\alpha, \beta \in \mathbb{C}$ are the probability amplitudes, that must satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. The qubits can be represented as vectors in an abstract vector space, called Hilbert space. One possible basis in this space is

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{1.2}$$

and then a superposition in this basis, namely $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, would be represented as

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{1.3}$$

Obtaining information from these states is possible via measurements. However, it is impossible to obtain enough information from the state in order to be able of reconstructing it, or in other words both values α and β cannot be acquired. After measuring the state ϕ, it will collapse into one of the states $|0\rangle$ or $|1\rangle$. The result of this measurement will be $|0\rangle$ with probability $|\alpha|^2$, or $|1\rangle$ with probability $|\beta|^2$. Is important to state that because a measurement implies an interaction with the system, the quantum state prior to a measurement will be lost, and the new state will be determined by the results of the measurement.

An important property of quantum systems is the *no cloning theorem*, first demonstrated

by Wooters and Zurek [Wootters and Zurek, 1982]. This says that is impossible to copy a quantum state. This property that in principle appears as inconvenient since the classical view uses typically the resource of duplicating information, becomes a powerful tool if properly exploited, because you can avoid with certitude that the information in such a system could be ever fully copied.

1.1.1. Entangled states

It is possible to have systems of several qubits. For example two qubits, let's say $|\psi_1\rangle, |\psi_2\rangle$ can be in their corresponding 'ground' state $|0\rangle$. This can be represented as, in different equivalent representations,

$$|\psi_1\rangle \otimes |\psi_2\rangle = |0\rangle \otimes |0\rangle \tag{1.4}$$

$$|\psi_1\rangle|\psi_2\rangle = |0\rangle|0\rangle \tag{1.5}$$

$$|\psi_1\psi_2\rangle = |00\rangle \tag{1.6}$$

They could also be each one in a superposition state $\frac{|0+1\rangle}{\sqrt{2}}$. Here the state of the system is

$$|\psi_1\rangle \otimes |\psi_2\rangle = \frac{|0+1\rangle}{\sqrt{2}} \otimes \frac{|0+1\rangle}{\sqrt{2}} \tag{1.7}$$

$$|\psi_1\psi_2\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) \tag{1.8}$$

This kind of states, that can be rewritten as a tensor product of n qubits for a system of n elements (in the previous examples $n = 2$) is known as *separable state*. On the other hand some quantum states cannot be separated in such a way. They are called *entangled states*.

For the particular case of a two qubit system $|\Gamma\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$, where $\{a, b, c, d\} \in \mathbb{C}$ and $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$, it can be proved that if $ad \neq bc$, then $|\Gamma\rangle$ is an entangled state.

Some examples of entangled states are the *Bell states*,

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \qquad |\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{1.9}$$

$$|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \qquad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{1.10}$$

the *GHZ state* of n qubits (standing for Greenberger, Horne and Zeilinger),

$$|GHZ_n\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1 \ldots |0\rangle_n + |1\rangle_1 \ldots |1\rangle_n) \tag{1.11}$$

and the *W state* of n qubits

$$|W_n\rangle = \frac{1}{\sqrt{n}}(|1\rangle_1|0\rangle_2 \ldots |0\rangle_n + |0\rangle_1|1\rangle_2 \ldots |0\rangle_n + \ldots + |0\rangle_1|0\rangle_2 \ldots |1\rangle_n). \tag{1.12}$$

1.2. Quantum communication

The exploitation of quantum systems by two (or more) parties for information transfer purposes is called *quantum communication*. Entanglement in particular could be exploited by them. Given that they share the qubits from an entangled state, when one of them, let's say Alice, measures its corresponding qubit, the result that the other party(ies) will obtain is correlated with Alice's measurement. There are also some applications that don't require a shared entanglement source, like the BB84 protocol exposed in section 1.3.1.

To implement quantum communication applications as well as quantum circuits, an opti-

cal approach was selected due to the experience of the research group where this project was developed.

For the production of the raw qubit material in the optical approach (the photons), a laser is typically used. Critical characteristics at the moment of selecting one laser are the output wavelength and repetition rate, if it's not from the continuous wave type. The wavelength will determine constraints on all the elements that follow the laser: mirrors, crystals, fibres, detectors.

The repetition rate determines how fast experiments can be performed. In spite of this, the control of these experiences becomes unmanageable when the repetition rate is too high. Another related issue is that detectors typically have a period of time where they are unable to detect a new photon after detection; this period is known as *dead time*. The detection systems will be discussed in chapter 2. Even more critical, the electronics that process all the information put a limit on how fast the experiments can run. This is the subject of chapter 3.

To keep track of a photon, it is handy to work with the *sync out* signal provided by some lasers. It provides the starting time of a single experience. So it is desirable to have a *sync out* signal from the photon source to be used, if some type of control or time processing is wanted.

Because qubits are realized by single photons in conventional schemes, weak pulses from lasers are often used. By attenuating the power of a laser, a single photon source can in principle be obtained. Because of its probabilistic nature, it can also generate several photons at the same time, or none of them with a certain probability.

Parametric down conversion is another method used to generate single photons. This process occurs in non-linear crystals and converts a photon of one wavelength to two photons with different wavelengths, conserving energy and momentum. Because both

output photons are entangled, when one of them is detected the other has been certainly created.

1.2.1. Polarization encoding

A photon property easily used for quantum information tasks is its polarization. Polarizers block one component of the electric field and transmit the other. From an observer's coordinate system, the output polarization can be horizontal, vertical, or any other position. It can represent the quantum state of single photons, where $|H\rangle$, $|V\rangle$ correspond to the preferred orthogonal positions, the so-called computational basis. One state can be represented as

$$|\psi\rangle = \alpha|H\rangle + \beta|V\rangle \tag{1.13}$$

where $|\alpha|^2 + |\beta|^2 = 1$. Polarizer beam splitters can separate these two polarizations in different paths. Actually they can separate any two orthogonal polarizations depending on its alignment. To modify the polarization state of photons, wave plates are used.

These elements and their behaviour are described by matrices, representing their transformations. This description is known as Jones formalism or calculus. Some examples are shown in table 1.1 [Saleh and Teich, 1991]. Thanks to the case of operating over a polarization qubit, this encoding type is mainly favoured for quantum computation experiences.

1.2.2. Time-bin encoding

When a photon enters a Mach-Zender interferometer, it can go through the short arm or the long arm, from a probabilistic point of view. In a quantum scale description, the photon goes by both arms at the same time. It is necessary to have a photon generated

Table 1.1 Jones' matrices for some polarization elements

Linear polarizer along x		$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$
Linear polarizer at angle θ		$\begin{pmatrix} \cos^2\theta & \sin\theta\cos\theta \\ \sin\theta\cos\theta & \sin^2\theta \end{pmatrix}$
Wave retarder	Quarter wave plate: $\Gamma = \frac{\pi}{2}$ Half wave plate: $\Gamma = \pi$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{-i\Gamma} \end{pmatrix}$
Polarization rotator		$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

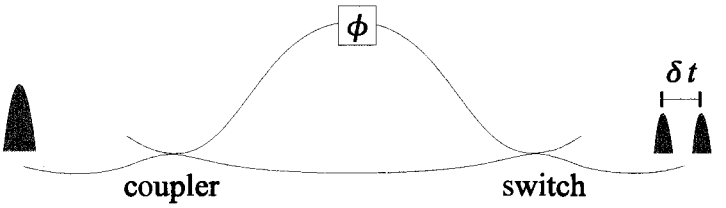


Figure 1.1 fibre interferometer for time-bin encoding, as introduced by [Brendel et al., 1999].

with a short pulse duration compared to the length difference between the arms of the interferometer [Brendel et al., 1999].

The basis used is such that when the photon passes by the short arm, it is in the $|s\rangle$ state, and for the long case its state is $|l\rangle$. Then, a qubit state in time-bin encoding is

$$|\psi\rangle = \alpha|s\rangle + \beta|l\rangle \tag{1.14}$$

Because fibres affect the polarization of photons on it, polarization encoding is not ideal for communication experiments. As an alternative time-bin can be implemented over fibres, since the interferometer can be fabricated with them. However, making computations in time-bin is currently unpractical, as there is no easy way to perform arbitrary unitary transforms on such a basis.

1.3. Quantum Protocols

A protocol is a set of rules that have to be followed in a certain order to achieve a defined goal. For example, to get money from an ATM a protocol between the machine and the user has to be followed: insert the card, type the password, etc. To register the executed transaction, the ATM has a protocol to communicate with a central database in a secure way.

Quantum protocols are a set of rules applied on quantum states, requiring a qubit source or even an entanglement source in some cases. These protocols become of interest when they can make something better than a classical version of them; more efficient answers, or eventually an answer that classically cannot be found.

When quantum cryptography is mentioned, the promise of a secure cryptography scheme comes to mind. Actually, what quantum mechanical properties offer is a way to create random keys and share them securely between two parties (Alice and Bob). This private random bit sequence can then be used to encrypt a classical message to be sent between them. One example of this type of protocol is exposed in section 1.3.1.

In cases two users prefer (or need) to transfer quantum information securely instead of classical information, they can use *quantum teleportation* [Gisin et al., 2002], provided that they share a prior entangled state, as explained in section 1.3.2.

1.3.1. BB84 protocol

The first quantum key distribution protocol is known as BB84, as the authors' initials and its publication year [Bennett and Brassard, 1984]. This requires the capability of creating a qubit, encoding it, sending it (Alice's side), receive it, measure it (Bob's side) and a classical communication channel for post-processing tasks between both parties.

The feasibility of each step depends highly on the implementation scheme and the type of encoding chosen.

The protocol goes as follows. Alice has to generate a couple of random classical bits: the first one will be the bit sent to Bob (x), and the second determines the basis in which Alice will encode the bit (b). The choice for the encoding basis can be anyone agreed between the two parties, with the condition that the two bases are orthogonal between them. For example, they can choose the rectangular and diagonal basis; for the rectangular basis a classical 0 is encoded in a $|0\rangle$ state and classical 1 in $|1\rangle$; for the diagonal basis a classical 0 is encoded in a $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state, while a classical 1 is in a $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ state.

Alice prepares and sends to Bob the state corresponding to her random pair of bits according to the previous rules. Bob then measures the received state in one of the basis (rectangular or diagonal, for example) in a random way, independent of Alice behaviour. Notice that if Bob chooses the same basis that Alice selected to encode the bit, the measured value will be the bit sent by Alice. If the opposite occurs, i.e. both choose a different but orthogonal basis, Bob will learn nothing about the state. This happens because he will measure 0 or 1 with probability $1/2$ in those cases.

To complete the protocol, Bob talks to Alice using a classical communication channel, and asks for each of the selected bases. If they have selected the same one, they keep the corresponding bit as part of their key. If they didn't, the bit is just discarded. They repeat this procedure as many times as they want, until they obtain the desired amount of shared bits.

There exist some variations of this protocol to make sure that a possible eavesdropper hadn't access to their quantum channel. One typical procedure includes the verification that some of the bits they claim to share are equal. This assures that nobody had modified

the information, since a third party in order to obtain information has to measure in one random basis (possibly a different one from the two chosen by Alice and Bob) and then resend the data, biased on its own selection of a basis. Remember the no-cloning theorem [Wootters and Zurek, 1982] that prevents a third party to copy the state, save a copy for himself and measure it after hearing the classical conversation.

1.3.2. Quantum teleportation

Thanks to entanglement, Alice and Bob can transfer between them a quantum state. Their interest could be to exchange the result of a quantum operation that only Alice is able to make, but Bob is the one who is interested in its output. The procedure known as *quantum teleportation* [Bennett et al., 1993] will permit them to achieve this. To do it they need 3 qubits, where two of them are entangled and shared between them, and the third one contains the quantum state to be transferred.

Before entering on the schematic description of the quantum teleportation procedure (section 1.3.2.2), is helpful to introduce the concept of quantum circuits (section 1.3.2.1).

1.3.2.1. Quantum circuits

Qubits can be altered. The second postulate of quantum mechanics states that:

> "The evolution of a closed quantum system is described by a unitary transformation" [Nielsen and Chuang, 2000]

Then, a modification to these states can only be done by unitary transformations $\hat{U}$ (where $\hat{U}^{-1} = \hat{U}^{\dagger}$) in the Hilbert space. Qubits can also be transferred and measured. The joint of these actions on qubits form what is called a *quantum circuit.*

An arbitrary transformation $\hat{U}$ applied to a state $|\psi_1\rangle$, such that its output $|\psi_2\rangle = \hat{U}|\psi_1\rangle$ is represented as

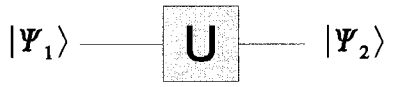


Figure 1.2 Arbitrary unitary gate.

The Walsh-Hadamard transformation $\hat{H}$, usually referred only as Hadamard, converts a $|0\rangle$ into a $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and a $|1\rangle$ into a $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Using the basis from equation (1.2), the $\hat{H}$ transform is represented as

$$\hat{H} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{1.15}$$

Its circuit application is represented as

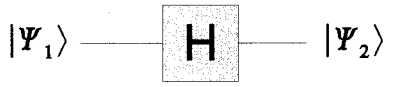


Figure 1.3 Hadamard gate.

Other useful one qubit gates are the negation $\hat{N}$ and the phase flip $\hat{P}$. They are given by

$$\hat{N} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \hat{P} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.16}$$

where for the negation gate $\hat{N}|0\rangle = |1\rangle$, $\hat{N}|1\rangle = |0\rangle$, and for the phase flip gate $\hat{P}|0\rangle = |0\rangle$, $\hat{P}|1\rangle = -|1\rangle$.

Another important transformation is the controlled-not gate, or just c-not, noted by $\hat{\oplus}$. It is a two qubit gate which changes the state of the second qubit (target qubit) from

$|0\rangle$ to $|1\rangle$ or from $|1\rangle$ to $|0\rangle$ only if the first qubit (control qubit) is in the state $|1\rangle$. Or, if $\{x, y\} \in \{0, 1\}$ then $\hat{\oplus}(|x\rangle_c|y\rangle_t) = |x\rangle_c|x \oplus y\rangle_t$. The corresponding matrix to this transformation in the $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ basis is

$$\hat{\oplus} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.17}$$

and the diagram which represents it is

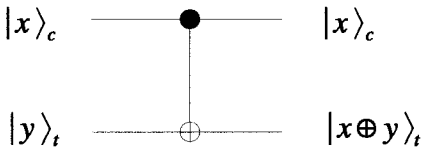


Figure 1.4 C-not gate.

1.3.2.2. Quantum teleportation procedure

The two interested parties, Alice and Bob, start by generating a pair of entangled qubits, a $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$, one of the Bell states from equation (1.9). Alice and Bob each keep one of these qubits, and therefore share entanglement. This state can be made from two qubits at $|0\rangle$, an $\hat{H}$ gate followed by a c-not, as shown in the first part of figure 1.5.

Then, Alice takes the state to be tranferred $|\Gamma\rangle$ and applies the next transformation between $|\Gamma\rangle$ and her part of the entangled state as indicated in figure 1.5.

At this moment, the state of the circuit is the following

$$|\zeta\rangle = \frac{1}{2}|00\rangle|\Gamma\rangle + \frac{1}{2}|01\rangle\hat{N}|\Gamma\rangle + \frac{1}{2}|10\rangle\hat{P}|\Gamma\rangle + \frac{1}{2}|11\rangle\hat{N}\hat{P}|\Gamma\rangle \tag{1.18}$$

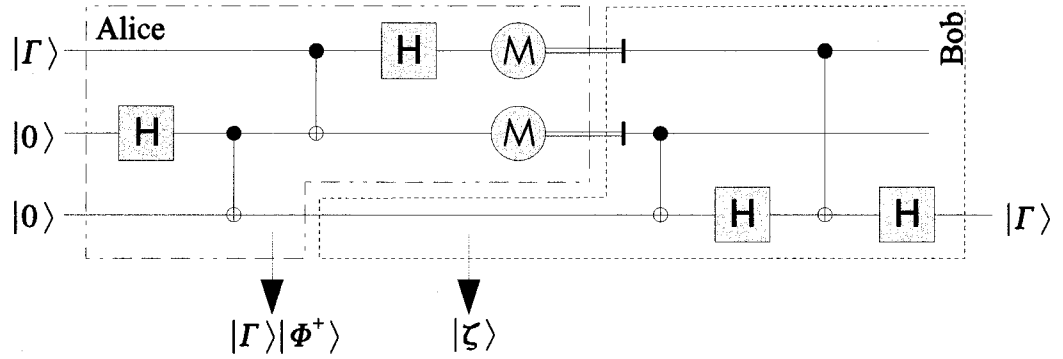


Figure 1.5 Quantum teleportation circuit. Alice and Bob share an entangled state $|\Phi^+\rangle$, which allows them to transfer the $|\Gamma\rangle$ state from Alice to Bob.

where Bob's state, the third qubit, is related to the tranferred state $|\Gamma\rangle$ except for $\hat{N}$ and/or $\hat{P}$ transformations; they were defined in equation (1.16). Alice then measures her qubits, immediatly collapsing Bob's state to one of the four options. She communicates via a classical channel the result of the measurement, so Bob can apply the correct transformation to recover $|\Gamma\rangle$ on his side.

1.3.3. Quantum coin tossing

Alice and Bob want to throw a coin and decide on a winner from this result. They are not in the same place. If one of them wants to cheat, he or she will succeed in a classical scenario every time. Nevertheless, in the quantum scenario protocols have been proposed to avoid that the cheater wins all the time. It has been demonstrated [Kitaev, 2002] that in the best case (from the honest person point of view) the cheater will succeed with a probability of about 71%.

One of these protocols was introduced in the same paper as the quantum key distribution scheme presented in section 1.3.1 [Bennett and Brassard, 1984]. Here is exposed the simplified version described recently [Berlín et al., 2008b]. Alice prepares one of four states at random, corresponding to two random bits like in the key distribution protocol:

$|0\rangle$, $|1\rangle$ for $b = 0$ or $|+\rangle$, $|-\rangle$ for $b = 1$. Then she sends this qubit to Bob.

Bob replies to Alice with a random bit g. Then he measures the received qubit in a random basis, namely Bob's $\tilde{b}$ with the same convention as Alice's b. His result will be $\tilde{x}$.

Alice declares her values for b and x. In the case that Bob guessed the basis ($b = \tilde{b}$) and the value measured doesn't correspond to the one sent by Alice ($x \neq \tilde{x}$), Bob recognizes that Alice is cheating. If Bob continues with the protocol, the coin value will be $b \oplus g$.

For this protocol Bob cannot cheat. Alice could cheat by lying over the value of b she used, or by sending a different state from the agreed four possibilities.

A newer version of the protocol [Berlín et al., 2008a] does not require a quantum memory. For the experimental implementation of this protocol, random numbers are required. If time-bin encoding is selected, depending on the selected basis a phase has to be applied in one of an interferometer's arm; this happens at encoding and decoding stages. Consequently, driving phase modulators is required. Single photon measurement at the end of the process is also required. All the above tasks can be done, with the support of the products of this project.

1.4. Experimental electronic requirements

Now that the fundations behind this work are explained, it is possible to talk about experimental requirements that can be addressed from the electronics.

Timing information of photon generation (laser fast photo diodes) and photon detection (single photon detectors) can be used to synchronize the process chain, which is useful to determine correlations between emitted photons and their measured counterparts. This

requires that the system receives different kind of analog signals and treats them equally. An analog to digital conversion stage scheme is presented at section 2.2.

For timing issues, an external clock is employed as a master clock for the whole system. It can be provided also by an electronic system. Recording this type of events with the timing information is used for this task. This procedure is presented in section 3.4.3.

To determine the actual state of a quantum state, measurements in different basis are done. Frequently coincidence (or anticoincidence) detection is required. This is discussed in section 3.4.2.

Coin tossing protocols even in its quantum version require the generation of classical random bits (section 1.3.3). Typically quantum communication protocols make use of this randomness source. This is possible to address with an electronic system, which creates a pseudo-random sequence of bits following a procedure exposed in section 3.4.4.

For the implementation of an interferometer designed for time-bin encoding (see section 1.2.2), it is necessary to control a phase modulator. This modulator, as discussed in section 4.1.1, is gated via a periodic electrical signal which depending on its voltage level indicates the phase that has to be applied to the optical signal.

CHAPTER 2

PHOTON DETECTION

As stated, detection of photons is a key step in experiments in optical quantum information, whatever encoding is been used. Even more critical, at the quantum level it is important to be able to detect a single photon. In some cases it is desirable to determine how many photons arrive, to facilitate distinction between the arrival of a single photon or a two or three photons event.

A wide variety of single photon detectors exist in the market. Each of them is designed to function for a certain wavelength range. Hence the choice of photon detectors is relevant when designing an optical experiment. This implies that for different setups, the detectors used are usually different.

This difference between detectors lies not only on its design, but also in its working principle and then, in the output that can be acquired for collecting data. There are several characteristics that differ, depending on the selection, such as efficiency, cost, temperature range operation and portability.

In section 2.1 several photon detection mechanisms are explored, putting a special emphasis on *avalanche photodiodes* (APDs), being the choice for our experiments. Dealing with the outputs of these detectors is considered in section 2.2. The proposal of a circuit which unifies the diversity of electrical signal types is exhibited in section 2.3, closing this chapter.

2.1. Photon detection mechanisms

Semiconductor APDs, semiconductor quantum dot detectors and superconducting detectors [Engel et al., 2004] are today's most discussed types of single photon detectors. Among them, the APDs are the most used since the other two types (quantum dot and superconducting ones) are recent technologies, which remain for the moment in an experimental stage.

The principle of operation for any photon detector is to generate or to switch on an electrical current whenever a photon arrives on the device. To succeed at the single photon level, the scheme has to be such that a single event produces a considerable flow of electrons so they can be measured. This macroscopic measurable event is known as a 'click'. In practice, not every photon that arrives to the device is able to produce a click, and therefore cannot be measured. The concept of *quantum efficiency* η arises as the following ratio

$$\eta = \frac{\text{detected photons}}{\text{incident photons}}. \tag{2.1}$$

It takes into account every effect that prevents a photon to generate a click in the device. η could be taken as the probability detection of a photon, since $0 \leq \eta \leq 1$. Quantum efficiency η has the property that for a given device, it changes with the photon wavelength λ. Curves of η vs λ are typically provided with the photon detector's product specifications.

The temperature of operation is a factor to take into account. APDs work typically between -50 °C and room temperatures, which can be controlled by using Peltier coolers. On the other hand, quantum dot detectors [Komiyama et al., 2000] and superconducting detectors [Engel et al., 2004, Rosfjord et al., 2006] operate at temperatures of a few or tens of Kelvin. As a result they have to be operated in a cryostat.

Dark counts is an important and undesired characteristic of photon detectors. The dark-count rate is the number of clicks per second when no photons are incident on the detector. Spontaneous decays of electrons in semi-conducting detectors can generate a dark count. Also, residual electrons from previous clicks could start a new process that emulates detection, while no-photon arrives; this phenomenon is known as *afterpulsing*.

After detection occurs, the system needs some finite time to recover. This interval is the *dead time*. Notice that during this period the device doesn't have the capacity to measure any arriving photon. This is a technical limitation for experiments that require detecting every single event at a fast rate.

Given these generalities of single photon detectors, the particular case of semiconductor APDs is now presented.

2.1.1. Used photodetectors: Avalanche Photodiodes

The operation principle of an avalanche photodiode (APD) is to convert the incident photon into a cascade of moving carrier pairs (electron-hole pairs) in a semiconductor p-n junction. It consists of a photodiode with a high reverse bias, which makes the carriers accelerate, with enough energy to excite new carriers in a process called *impact ionization* [Saleh and Teich, 1991]. When one photon arrives, it is absorbed and generates a carrier pair (figure 2.1). The electron (hole) is accelerated by the strong electric field in the semiconductor. If the energy acquired by the electron before it collides with other one in the semiconductor is enough to have a kinetic energy greater than the gap energy E_g, it will liberate another electron. These two electrons continue gaining energy from the electric field, and will collide with other electrons, and so on.

Typical values of operation wavelengths for APDs are $400 \text{ nm} < \lambda < 900 \text{ nm}$ for those made in Si, and $1\ \mu\text{m} < \lambda < 1.7\ \mu\text{m}$ for those made in Ga, InGaAs and InGaAsP.

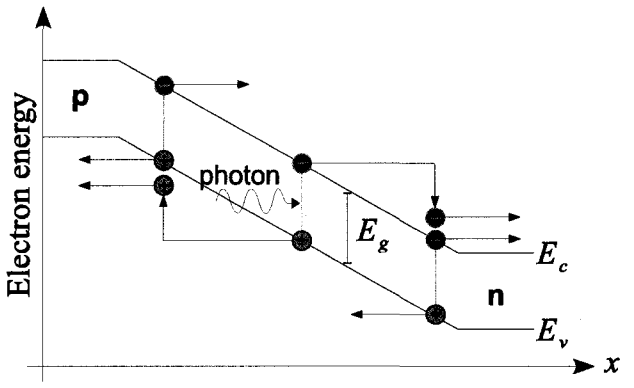


Figure 2.1 Avalanche process in APDs.

Silicon detectors are often used for detecting free-space beams, favouring polarization encoding experiments. In a particular quantum information experiment for which this work was destined, a Ti:sapphire laser at 720 nm is used to generate 360 nm photons, which create both 680 nm and 765 nm photons; they are collected by using silicon detectors.

Another experiment taken into account by this work, is a quantum communication implementation on optical fibre that uses time-bin encoding with photons at 1550 nm. For this particular case, because of the photon's wavelength, InGaAs detectors are used. In general, InGaAs detectors are used in experiments that uses optical fibres, since they work in the transmission window of 1500 nm as well as in the 1300 nm.

To achieve single photon resolution in the detection process, an APD is operated in *Geiger mode*, whereas the applied bias voltage V_E is larger than the breakdown voltage. The problem with this situation is that the current diverges after a click, to the point of exceeding the limit where the high current obtained destroys the device [Karlsson et al., 1999].

Therefore, there must be a circuit controlling the avalanche process to avoid permanent damage on the APD. These circuits are called *quenching circuits*, and they come in sev-

eral schemes, such as passive quenching, active quenching and gated quenching circuits.

The quantum efficiency defined in equation (2.1), depends on the probability that a photon is absorbed in the semiconductor layer (absorption efficiency), the probability that the carrier generated by the photon starts an avalanche process (trigger probability) and on the optical coupling efficiency of the light to the device. Quantum efficiency greater than 10% can be achieved in InGaAs detectors and about 70% for silicon detectors.

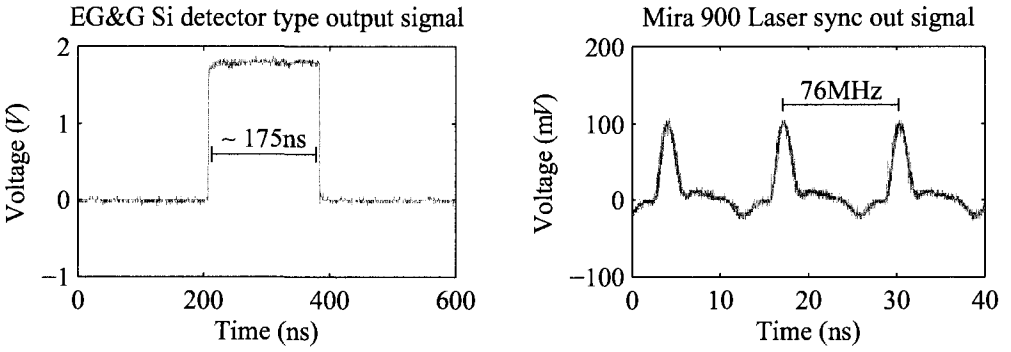
In terms of dark counts, good silicon single photon detectors exhibit between 10 to 100 dark counts per second, and for those made on InGaAs or InP this rate goes in the range 100 to 1000 dark counts per second in the best conditions.

2.2. Signal conversion and standardization

Commercial electronic devices often come with circuits that convert their original analog outputs into signals compatible with other electronic systems. The output is often digital, using one of the existing standards like TTL. However this is not always the case. In this section this inconvenience is addressed.

In this project three types of electrical signals have to be detected and standardized such that a complementary circuit may read them and treat them equally. They are

- Silicon single photon counting module. TTL signal. Amplitude: 1.8 V. Pulse duration: 175 ns. See figure 2.2(a).
- InGaAs photodiode. Analog signal. Amplitude: 180 mV. Pulse duration: 7 ns.
- Laser fast diode output (sync out). Analog signal. Amplitude: 100 mV. Pulse duration: 2.5 ns. Repetition rate: 76 MHz. See figure 2.2(b).



(a) Typical output signal of a EG&G silicon single photon counting module.

(b) Sync. output signal from a Mira 900 pulsed laser.

Figure 2.2 Examples of input signals for the electronic system.

The solution for detecting input electrical signals is designed to be universal. The idea is to compare the signal, without regard to its form, with a reference voltage as shown in figure 2.3. When the signal V_{in} is higher than a reference level V_{ref}, the circuit should convert it to a digital '1' high level; a digital '0' low level should be obtained in the contrary.

The response of this circuit has to be fast enough to detect changes in signals that last about 3 ns in the so called high level. Also it has to detect signal amplitudes as low as 100 mV, meaning that V_{ref} has to be approximately 40 mV.

The proposed solution is to use a driver/receiver for differential signalling. Instead of giving the device a differential pair signal as input, it will get the voltages V_{in} and V_{ref}. Its output gives a differential pair of amplitude defined by the device characteristics, which has as possible values only the two required ones, high and low, '0' and '1'.

A Micrel® *Ultra-precision differential* 800 *mV LVPECL line driver/receiver with internal termination* (SY58601U) was selected for this purpose. Its maximum operating frequency is 5GHz (or 5Gbps). Its output rise/fall time (t_r, t_f) covers the range be-

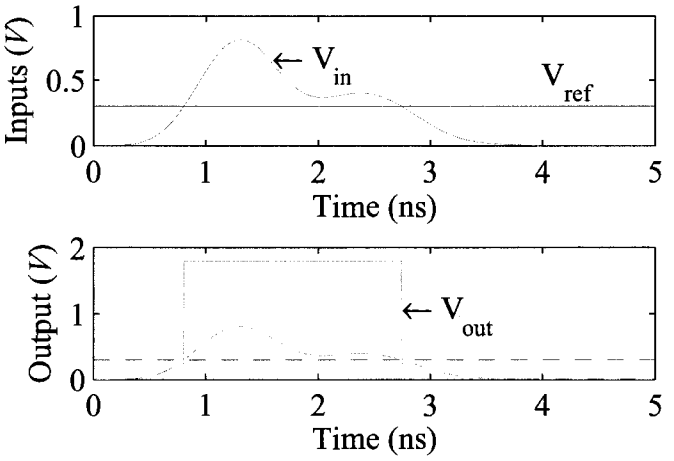


Figure 2.3 Principle of operation for the conversion of signals to TTL.

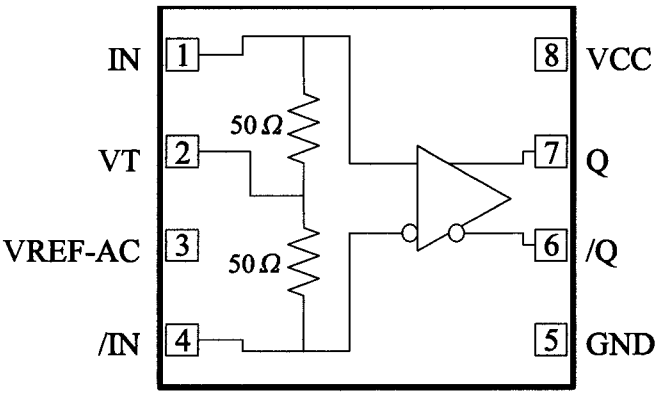


Figure 2.4 Micrel SY58601U internal schematic.

tween 25 ps and 90 ps. Its output voltage swing is typically 800 mV for each pin, being 1600 mV the voltage swing for the differential output. When used within its design specifications, input swing accepted can be as low as 100 mV.

The device outlook, as shown in figure 2.4, is used as follows: pin IN is connected to the detector signal V_{in}; pin VT is connected to an extern voltage V_T; pin /IN is connected to V_{ref}; pin VREF-AC remains unused.

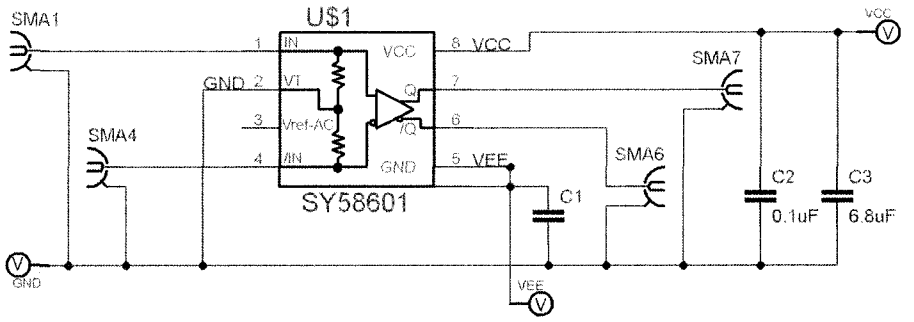


Figure 2.5 Schematic of Micrel SY58601U evaluation board. DC-coupled configuration.

Table 2.1 Configuration cases for Micrel SY58601U DC-coupled evaluation board tests.

Case	**V_{in}**	**V_{ref}**	**V_{CC}**	**V_{EE}**
First configuration				
	TTL signal, amplitude $>$ 400 mV	variable	2.5 V	0 V
Second configuration				
Scenario 1	TTL signal, 100 mV amplitude	70 mV	2.00 V	-1.21 V
Scenario 2	Mira900 laser sync out signal	0 mV	2.00 V	-1.21 V
Scenario 3	Mira900 laser sync out signal	40 mV	2.00 V	-1.21 V

2.2.1. Test results with evaluation board

Using an evaluation board provided by Micrel (the SY58601U device producer), and modified for DC coupled operation, the chip was tested to confirm its performance and the usefulness to compare voltages. The schematic of the used configuration is shown in figure 2.5.

It was initially tested with a supply of $V_{CC} = 2.5$ V and $V_{EE} = 0$ V (first configuration, in table 2.1). As input a TTL signal of variable amplitude created with a signal generator was used. The reference level was provided by a variable voltage divider. Its operation limit appeared when the TTL signal amplitude was reduced to less than $V_{in} \approx 0.4$ V. With this supply configuration, the signals of 100mV definitely could not be recognized.

A second configuration was tested, where the supply voltages were $V_{CC} = 2.00$ V and $V_{EE} = -1.21$ V (second configuration in table 2.1). In this one, TTL signals of amplitude

100 mV were correctly detected, as shown in scenario 1 (figure 2.6(a)). The case of the sync out signal from the Mira 900 laser was tested, as the limit case of signals that would be detected by the system. In scenarios 2 (figure 2.6(b)) and 3 (figure 2.6(c)) is shown that for a $V_{\text{ref}} = 0$ V and $V_{\text{ref}} = 40$ mV respectively, the outputs Q and $/Q$ behave such that

$$\text{if} \quad V_{\text{in}} > V_{\text{ref}} \quad \text{then} \quad Q = Q_{\text{high}} > 0 \quad \text{and} \quad /Q = /Q_{\text{high}} < 0 \quad \text{(high case)} \tag{2.2}$$

$$\text{if} \quad V_{\text{in}} < V_{\text{ref}} \quad \text{then} \quad Q = Q_{\text{low}} < 0 \quad \text{and} \quad /Q = /Q_{\text{low}} > 0 \quad \text{(low case)} \tag{2.3}$$

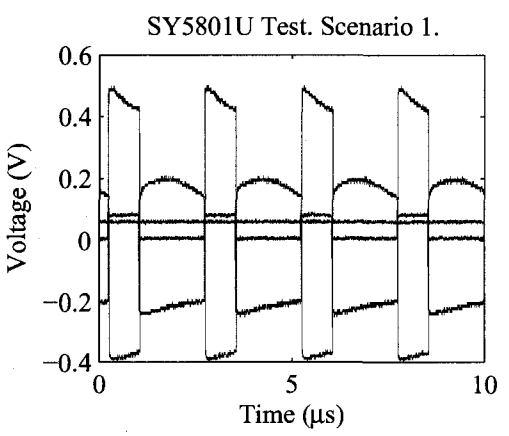
where the high case correspond to the logic '1' value, and the low case to the logic '0'. A good differential pair output happens when $Q = -/Q$ and $Q_{\text{high}} = -Q_{\text{low}}$, which is closely obtained in scenario 2. It can be checked that the first condition, $Q = -/Q$, always occurs. To achieve the second condition, $Q_{\text{high}} = -Q_{\text{low}}$, the V_T value should be adjusted.

2.3. The input circuit

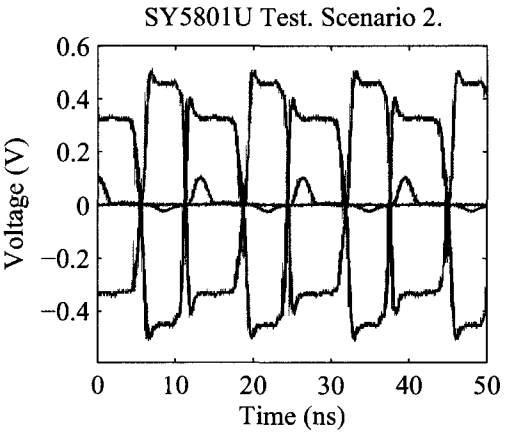
After testing the evaluation board, a custom designed PCB is made. This new circuit board has the capability to drive eight signals at the same time. The differential output is used by the ZestSC2 card, which is presented in section 3.2.

To design the circuits made in this project, the software Eagle 5.2.0 Standard edition was used. The purpose of this Printed Circuit Board (PCB) is to convert eight signals, using the scheme described in section 2.2, and to provide as output the same number of differential pairs, done in digital format.

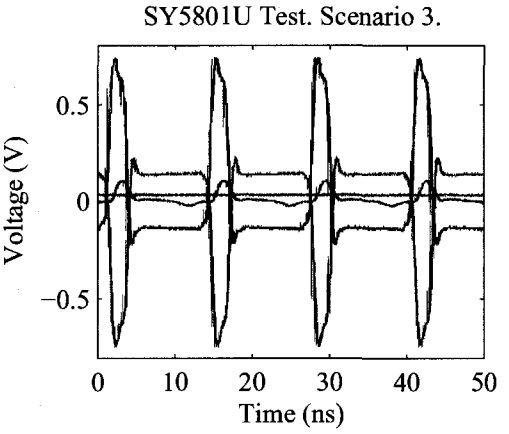
This PCB is designed to work with a ZestSC2 FPGA USB card, from Orange Tree technologies. The FPGA characteristics are treated in chapter 3. A short description of



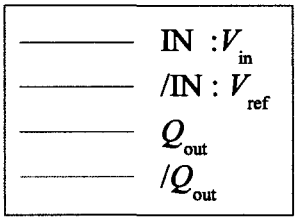
(a) Input V_{in}: TTL signal of 100 mV. V_{ref} = 70 mV DC signal.



(b) Input V_{in}: laser sync out signal. $V_{\text{ref}} = 0$ V DC signal.



(c) Input V_{in}: laser sync out signal. V_{ref} = 40mV DC signal.



(d) Legend for figures 2.6(a), 2.6(b) and 2.6(c).

Figure 2.6 Micrel SY58601U DC-coupled evaluation board tests. Scenarios in second configuration: $V_{CC} = 2.00$ V and $V_{EE} = -1.21$ V.

this card's characteristics is introduced in section 2.3.1.

2.3.1. PCB schematic

The supply voltage of the designed PCB is $V_{CCO} = 3.3$ V, the working voltage of Micrel SY58601U. Any signal applied as an input must not be higher than the supply voltage; it would cause damage to the Micrel SY58801U device. The FPGA's $V_{FPGA\text{-}CCO}$ is set at 3.3 V for compatibility with the PCB. Nevertheless the supply voltage V_{CCO} is extern. There exist the option to shortcut this two voltages, by using a shunt in the connector SV1 (2x1 pin header, see figure 2.7).

The VT voltages are supplied externally also. The reference voltages V_{ref} are produced by the circuit, via eight voltage dividers graduated with potentiometers. They go from GND to V_{CCO}. To maintain the V_{ref} values protected from circuit currents, unity-gain stable power amplifiers were used. In particular, LM4880 devices are employed. In the case that VT is desired to be the same as its corresponding V_{ref}, instead of supplying VT externally, a shunt can be placed in the corresponding connector SV2 (8x2 pin header).

For acquiring V_{in} signals, because the possibility of having high frequency signals (few GHz), SubMiniature version A (SMA) connectors were selected instead of Bayonet Neill Concelman (BNC) ones. The outputs are driven to a 32x2 pin header connector, to fit directly in the J8 FPGA header of the ZestSC2 card.

Several bypass capacitances were placed, as can be seen in figure 2.7. For the Micrel SY58601U, close to the V_{CC} pin are one 10 nF ceramic capacitance per device and one 0.1 μF tantalum capacitance per couple of devices. Their inputs $/IN$ and VT have bypass 0.1 μF capacitances close to the Micrel device, since they are reference voltages. For the supply voltage, bypass capacitors in parallel 1000 μF$||$10 μF are placed next it. The LM4880 amplifier requires a 1 μF between its bypass pin and ground. Next to the

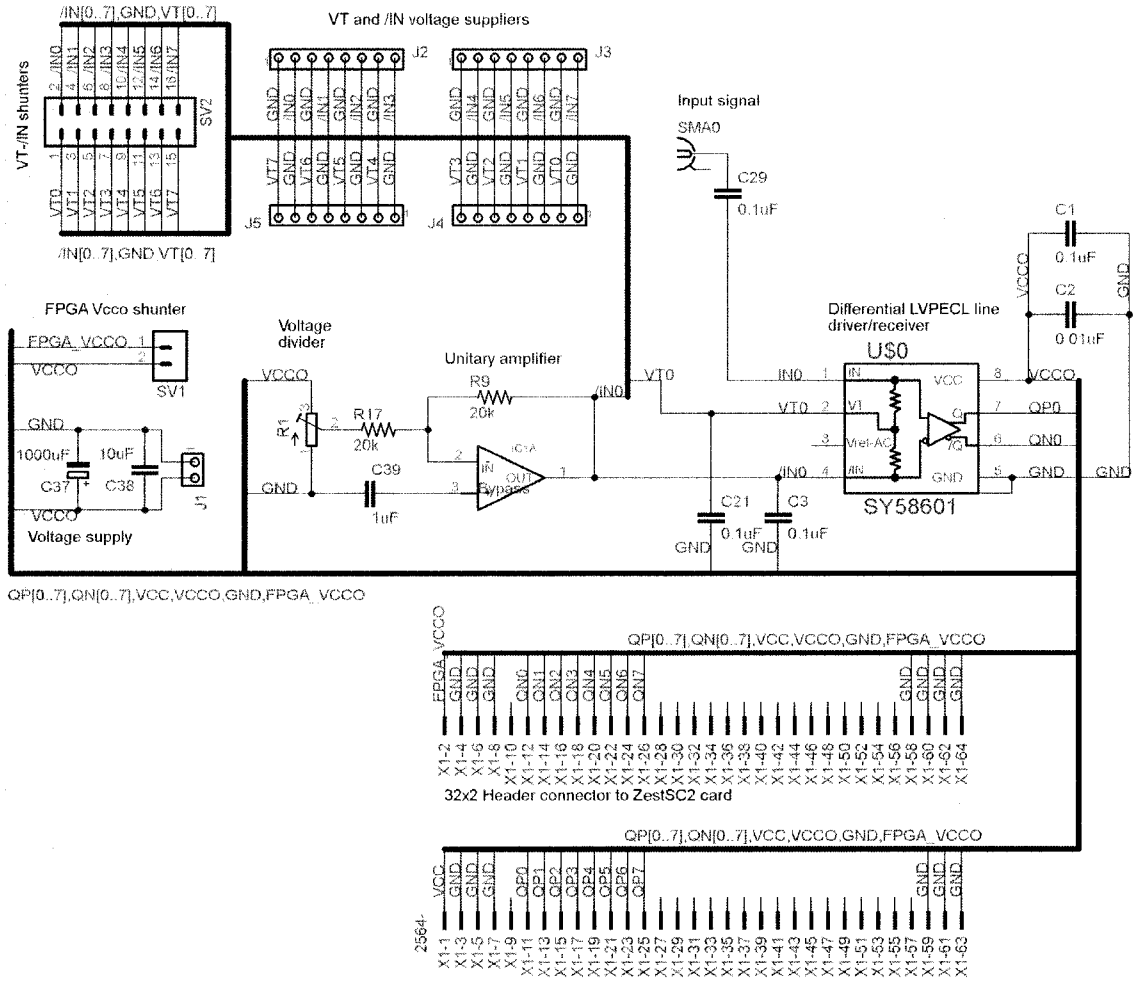


Figure 2.7 Schematic of designed input PCB. The treatment circuit for a single signal is here depicted, which includes a voltage divider, a unitary amplifier, and a differential driver/reciever. The actual PCB drives eight of these signals.

Table 2.2 Components in the input circuit PCB.

Device	**Quantity**
Connectors	
SMA PCB horizontal connector	8
Connector header vertical, 2 positions. 0.100in	1
Connector header vertical, 8 positions. 0.100in	4
Pin header 2x1. 0.100in	1
Pin header 8x2. 0.100in	1
Pin header 32x2. 0.100in	1
Integrated circuits	
LM4880	4
Micrel SY58601U	8
Resistors and capacitances	
1 kΩ trimmer potentiometer Y	8
10 nF ceramic capacitance SMD 0603	8
0.1 μF tantalum capacitance SMD 1206	24
1 μF tantalum capacitance SMD 1206	4
10 μF tantalum capacitance SMD 1206	1
1000 μF electrolitic capacitance	1

signal SMA connectors are placed optional 0.1 μF capacitances, as recommended by the producer in the interface applications.

In table 2.2 is the whole list of required elements for this PCB.

2.3.2. PCB layout

To make the layout for fabrication, the Micro Lead Frame of eight pins (MLF-8) package is added to the physical layout library; the Micrel SY58601U device uses this package. The specifications are given in the datasheet of the component.

A 4 layer PCB was used. The top and bottom layers are assigned to routing signals, while the second is dedicated to GND and the third to V_{CCO}. The routing was manually made for the case of sensitive signals, like the eight V_{in} from the SMA connectors, the Micrel

bypass capacitances and the Q, /Q output signals. In the case of the differential outputs, their lengths have to be as equal as possible, and also following a path as symmetrical as possible. The remaining signals were routed using the Eagle autorouter application.

50 Ω impedance was desired for the inputs and outputs of each Micrel component. To achieve this, the widths of the microstrip traces are specially calculated. Several formulas that give an approximation of PCB impedances exist. The used equation is done by the IPC document IPC-D-317A [Brooks, 1998]

$$Z_0 = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln\left(\frac{5.98H}{0.8W + T}\right)\Omega \tag{2.4}$$

where H is the height of the dielectric between the ground plane and the trace, W is the width of the trace (to be determined), T is the thickness of the trace and ε_r is the relative permittivity of the dielectric.

These characteristics are given by the company which produces the PCB. In this case Multifor Ltd., located in Dorval, QC, offers $H = 6\,\text{mil}(0.1524\,\text{mm})$ $T = 1.7\,\text{mil}(0.04318\,\text{mm})$ $\varepsilon_r = 4.3$ Giving as a result for $Z_0 = 50\Omega$, $W = 9.234\,\text{mil}(0.2345\,\text{mm})$. The GND paths were made with $W = 24\,\text{mil}$.

The connections layout for this circuit is given in the appendix I.1.

2.4. Chapter summary

A signal conversion system was developed. The necessity of reading analog signals of different kinds justify the creation of such a system; typical signals to be addressed in quantum optical experiments are silicon and InGaAs single photon detectors outputs, and synchronisation outputs from lasers.

The main component (Micrel SY58601U) was tested successfully for the conversion purpose, providing a differential output that can be read by an FPGA. Then a PCB proposal which is able to convert up to 8 signals with different characteristics was presented. These converted signals are the inputs of the information processor, based on an FPGA; this processing unit is the subject of the upcoming chapter 3. In particular, the PCB was specifically designed to work with a ZestSC2 card (see section 3.2).

CHAPTER 3

INFORMATION PROCESSOR AND APPLICATIONS: THE FPGA

Quantum information systems in general require treatment of information generated from measurements, from just counting clicks to recording time information of these events for post-processing and analysis. They may also make actions, like turning on a device or changing the phase applied by an electro-optical modulator. Any particular procedure that could arise in these experiences should be done by one electronic system which acts as a processor: managing resources, reading and sending information.

Due to the great variety of tasks to perform depending on the particular implemented set-up, a system capable of being reprogrammed is advantageous. This reduces the cost and increases the utility of such a system. The speed of this processor is another requirement to take into account. From Shannon's sampling theorem,

Theorem 3.1 *If a function $f(t)$ contains no frequencies higher than W cps (cycles per second), it is completely determined by giving its ordinates at a series of points spaced $1/2W$ seconds apart. [Shannon, 1949]*

In the worst case, the system has to be twice faster than the fastest signal to be detected. In one case, the limit signal can be the laser sync out signal, of $f = 76$ MHz, meaning that the lower operation frequency for a set-up using the laser sync out will be of $f_{\mathrm{CLK}} = 152$ MHz.

Another limit to the processor's frequency is the interferometer path length difference. The processor must be able to distinguish between a signal that passes through the long arm of an interferometer and the short one. The acquisition rate f_{CLK} determines the

extra time Δt that light could take to pass by the long arm, therefore the extra distance Δx that it has to traverse.

$$\Delta t > \frac{1}{2f_{\text{CLK}}} \tag{3.1}$$

Taking into account these considerations, a *Field-programmable gate array* (FPGA) is used in this project. They accomplish the reconfigurable requirement. The speed of FPGAs in the market ranges from a few MHz to 1GHz (a 1066 MHz commercial product was announced in June 2008); the faster an FPGA works, the less affordable it becomes. In particular, a ZestSC2 card from Orange Tree Technologies was used. It contains a Xilinx® Spartan-3 XC3S2000-4 FPGA, a Universal Serial Bus (USB) to communicate with a computer, as well as other components and features (see section 3.2). It works at $f_{\text{CLK}} = 48$ MHz, but other frequency clocks can be synthesized as explained in section 3.1.2.

3.1. Spartan-3 FPGA characteristics

The details of how an FPGA works and the justification of its uses are presented here. Particularities of the Spartan-3 XC3S2000-4 FG676 version from Xilinx® are also addressed. It is convenient to decompose the long name of the used card: Xilinx is the producer, Spartan-3 is the family, XC3S2000 is the device, -4 is the speed grade, and FG676 is the package type.

An FPGA is formed by a distribution of interconnected functional blocks, the fact that gives the second half of its name "gate array". The "field-programmable" label makes reference to the capability of programming these blocks and their connections. The conventional sorts of components that could be found in such an array are:

- Configurable Logic Blocks (CLBs) that implement the logic circuits and some

Table 3.1 Xilinx Spartan-3 XC3S2000-4 FG676 block components.

Component	Quantity
System gates	2M
Logic cells	46080
CLBs	80 rows × 64 columns = 5120
RAM Blocks	40
DCMs	4
Dedicated 18bit×18bit multipliers	40
User I/O	489
Differential I/O pairs	221

storage elements.

- Input/output blocks (IOBs) that control the flow of information between the inner circuits of the FPGA and its I/O pins.
- Random Access Memory (RAM) blocks to save data.
- Digital Clock Manager (DCM) blocks, which permit the generation of different clock signals.
- Dedicated algorithmic blocks to enhance the performance of particular tasks.

In the case of Spartan-3 chips, the dedicated algorithmic blocks included are multipliers. The amounts of blocks available in an XC3S2000 device are listed in table 3.1.

3.1.1. Configurable logic blocks

The main component of a configurable logic block (CLB) is a function generator which is based on a RAM. It is called a Look-Up Table (or LUT). It takes n binary input bits ($n = 4$ for the Spartan-3 family), and gives as a response a single bit. A logical function $f(x)$ can be implemented with LUTs: a given digital input $x_{\text{in}} \in \{0, 1, \ldots, 2^n - 1\}$ has

a single output $x_{\text{out}} = f(x_{\text{in}}) \in \{0, 1\}$ corresponding to the assigned value in the truth table programmed in the LUT.

Another important element to be contained in a CLB is a storage component. Flip-flops type D serve to save the output of LUTs and synchronise them within the system clock signal. Other helpful elements in a CLB are multiplexers, simple logic function as AND-gates, OR-gates, and XOR-gates.

3.1.2. Digital clock managers

A digital clock manager (DCM) provides the possibility to synthesise clocks with frequencies that are small integer multipliers and divisors of the system main clock frequency. In other words, the new frequency f_{CLKFX} in terms of the main clock frequency f_{CLKIN} is

$$f_{\text{CLKFX}} = f_{\text{CLKIN}} \times \frac{M}{D} \tag{3.2}$$

where $\{M, D\} \in \mathbb{N}$. More specifically $2 \leq M \leq 32$ and $1 \leq D \leq 32$. The ZestSC2 card provides $f_{\text{CLKIN}} = 48$ MHz to the FPGA. Nevertheless the achievable frequency is bounded for each device; in Spartan-3 family case, $18 \text{ MHz} < f_{\text{CLKFX}} < 210$ MHz [UG331, 2008, p.134].

Another utility of a DCM is the phase shifting of synthesized clocks. A clock signal can be obtained with four possible phase shifts: $0°$, $90°$, $180°$ or $270°$.

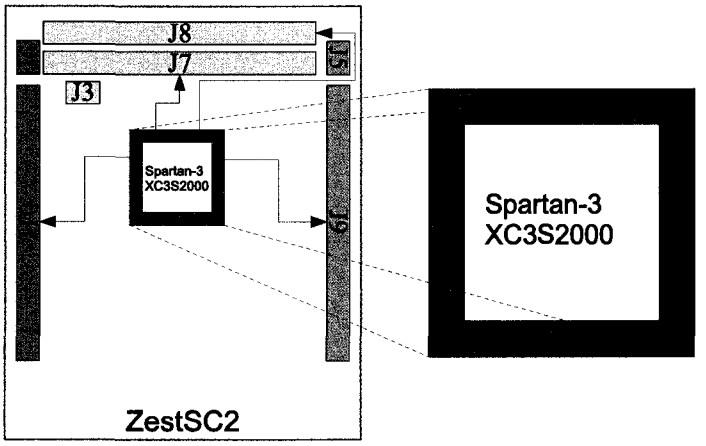


Figure 3.1 Input/output banks in XC3S2000 FPGA and their distribution in ZestSC2 card.

3.1.3. Regular and differential input/output ports

The FPGA supports several signalling standards both single-ended and differential ones. Some examples of supported single-ended standards are LVCMOS at 1.2 V, 1.5 V, 1.8 V, 2.5 V and 3.3 V, LVTLL at 3.3 V, and supported differential standards examples are LVDS and LVPECL.

The input/output blocks (IOBs) are distributed in 8 banks in a XC3S2000 device, as shown in figure 3.1. Each bank has an independent supply voltage V_{CCO}. In the case of the ZestSC2 card, the user has access to the banks 0, 1, 2 and 7. Also the user can choose the V_{CCO} of each of these banks. ZestSC2 connectors labelled J3, J5 and J6 serve for this purpose. J3 sets banks 0 and 1, where they always have the same V_{CCO}; J5 sets bank 2 V_{CCO} and J6 sets the one at bank 7. For further information refer to the ZestSC2 User Guide [Sweeney and Bowen, 2006].

Not every I/O pin is differential capable. To distinguish which of them have this property, it suffices to look at the *pin name* in the datasheet of Spartan-3 family [DS099, 2008], and it has to be in the format "Lxxy_#" where the 'L' indicates differential capability, 'y'

can be 'P' or 'N' referring to the positive or inverted signal of one differential pair, and '#' indicates the bank where the pins are located.

For a given package, FG676 in our case, each *pin name* has an associated *pin number*, which are one or two letters followed by one or two digits (i.e. A5, R22, AD15). When programming the FPGA, the pin number is the one that will be stated and not the pin name.

3.2. Other characteristics of ZestSC2 card

The ZestSC2 card produced by Orange Tree Technologies comes with a set of devices and software with the Spartan-3 XC3S2000-4 FPGA. They can be explored in the ZestSC2 user guide [Sweeney and Bowen, 2006]. Among them are a USB communication system, an SDRAM memory, a flash memory, light emitting diodes (LEDs) and software including a C library of functions, VHDL (Very-high-speed integrated circuits Hardware Description Language) and Verilog configuring examples for the FPGA, the ZestSC2 card driver, a user constraints file (UCF) and some executable utilities.

Power supply to the card can be provided via the USB connection or from an external 5 V outlet. For USB supply, the port must have 500 mA of total power available. However it is recommended to provide an external supply.

There are two ways to program the FPGA. When the card is connected, it proceeds to configure the FPGA from the flash memory content. It has to contain a valid configuration file; the flash memory can be programmed by using the utility *ProgFlash* included within the card documentation. The second way to configure the FPGA is using the USB port at any moment.

The USB communication between the FPGA and the computer is done by an interface

controller which includes a 16 bit streaming bus, an 8 bit data bus for registers, another 8 bit data bus for flags, and one signal for interrupt from the FPGA to the USB controller. This communication works at 48 MHz, providing a maximum streaming speed of 96 Mbytes per second.

The Synchronous Dynamic Random Access Memory (SDRAM) has a 16 bit data bus to communicate with the FPGA. Eight of the thirteen LEDs are available for programming; the other five are for status indication purposes. I/O ports provide access to 200 FPGA pins. They are distributed in 4 connectors, from J7 to J10, as shown in figure 3.1. Some of them are differential pair capable.

3.3. Software used

There are three tasks that are done using computer software, in order to operate an FPGA:

- Generation of configuration files
- Configuration itself
- Communication with a configured FPGA

Each of them requires a specific approach. For the description of FPGA's internal behaviour, VHDL code is used. It is possible to generate a configuration bit-stream file from this type of code using software provided by the producer of the FPGA. In this case Xilinx ISE was used. To make the configuration itself, executable files calling the bit-stream file are generated from C code. Microsoft Visual Studio was used. Finally for the communication between computer and FPGA, C functions can be used. For this purpose executables from C code can be used, or Dynamic Link Libraries (DLL) can be

created to call them from a *Graphical User Interface* (GUI) like National Instruments' LabVIEW.

3.3.1. Xilinx ISE and VHDL code

To dictate the FPGA how to configure its CLBs (described in section 3.1.1), a binary file is used. The software provided by the producer is necessary to generate this kind of file for a specific device of FPGA. For the Spartan-3 XC3S2000 device from Xilinx, the software offered is Xilinx ISE. As of the date of redaction of the present document, the ISE WebPack version (free downloadable version) was not available for the XC3S2000 device family; therefore no valid binary file could be generated with it. An ISE Foundation version (full version) is required. Since this software is being developed and tested continuously, it is strongly recommended to work with the latest version available to avoid software bugs.

The behaviour of the FPGA is written in VHDL language. This code is designed to describe hardware. Components can be described as blocks with an inherent behaviour, inputs and outputs. *Process components* define these functional behaviours, such as and-gates, or-gates, multiplexers, latches, flip-flops or counters. Any synchronous or asynchronous behaviour is depicted by them. On the other hand the architecture of the system is given by how these blocks are used and interconnected; this is described by *structure components*. Structure components may have inside several process components and other structure components as well.

Xilinx ISE software does the compilation of a VHDL project in three steps. First, the *synthesize* phase verifies VHDL code syntaxis and generates a corresponding Register Transfer Level (RTL) schematic. This one establishes with registers and combinational logic a circuit equivalent to the VHDL code. Then, the *implement design* stage checks

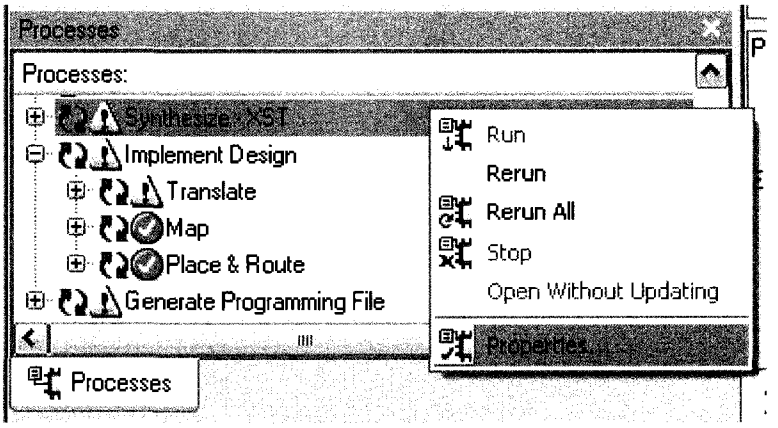


Figure 3.2 Xilinx ISE processes window. To modify a property, right click on the process and select the properties option. Then the properties window appears.

the availability of resources of a specific FPGA model and manages them to enable the implementation of the code. Finally the software proceeds to create the bit-stream file in the *generate programming file* step.

For the ZestSC2 card implementation over Xilinx ISE software, some changes are required [Sweeney and Bowen, 2006]. When selecting the project in the 'sources' window, the options shown in figure 3.2 appear at the end of the 'processes' window. Modification of their properties can be made by a right click on each of them and selecting the 'properties' option. The properties given in table 3.2 should be set. To see all the properties, check that the 'property display level' is set in 'advanced'.

3.3.2. FPGA configuration load and Visual studio

Once a bit-stream file (`file_name.bit`) is generated by the Xilinx ISE software, it has to be loaded on the FPGA to run it. As stated in section 3.2, configuration of the FPGA in the ZestSC2 card can be done by using the ProgFlash utility (provided with ZestSC2 documentation) or by using C functions. Here the C code approach is explained; Microsoft Visual Studio was used.

Table 3.2 Property changes in Xilinx ISE for a ZestSC2 card.

Category	Property name	Value
Synthesis – XST		
Xilinx Specific Options	Pack I/O registers into IOBs	Yes
Implement design → Translate		
Translate Properties	Allow unmatched LOC constraints	Yes
Implement design → Map		
Map Properties	Perform timing driven packing and placement	Yes
Map Properties	Allow logic optimisation across hierarchy	Yes
Generate Programming File		
Configuration Options	Unused IOB pins	Pull up
Startup options	Drive done pin high	Yes

The C file has to include the library `ZestSC2.h`. This library includes the function `ZestSC2ConfigureFromFile`, which configures the FPGA passing as parameter the name of the bit-stream file. The simplest code looks like the following lines

```
void main(void) {
  ZESTSC2_HANDLE Handle;          %variable declaration

  ZestSC2OpenCard(1, &Handle);  %Opens FPGA with CardID=1
  ZestSC2ConfigureFromFile(Handle, "file_name.bit");
  ZestSC2CloseCard(Handle);       %closes FPGA
}
```

where all the `ZestSC2` functions are included in the cards provider library. An executable file (`file_name.exe`) is compiled. When running the executable file, the corresponding bit file must be in the same folder. Also the ZestSC2 card must be connected to the computer, and already recognized by it; this means that the card's driver has to be previously installed.

Another option is to use the `ZestSC2LoadImage` function. With this function a bit-stream file is loaded in memory before actually programming it over the FPGA. This can

be useful when the decision of the bit-stream file to load is made during execution of a C program.

Data transfer between host computer and FPGA can be done with C functions. For streaming data tranfer, functions `ZestSC2ReadData` and `ZestSC2WriteData` exist. Functions `ZestSC2ReadRegister` and `ZestSC2WriteRegister` access a specific register in FPGA's internal RAM memory. The 8 bit flags bus is configured via a C function (`ZestSC2SetSignalDirection`), as well as the corresponding read and write functions (`ZestSC2ReadSignals` and `ZestSC2SetSignals`). Interruption signal from FPGA can be read with function `ZestSC2WaitForInterrupt`. Error handling functions also exist. However, it is a good idea to design the behaviour of the FPGA such that the LEDs indicate the status of the loaded program, since the error handler only covers some error cases.

Deeper explanations can be found in the card's user guide, as well as studying the five examples provided by Orange Tree Technologies. They have to be studied together with their corresponding VHDL code sources to understand them. These codes have already configured the communication systems between the FPGA and the other components in the ZestSC2 card (SDRAM, LEDs, USB).

3.3.3. Labview and DLL libraries

For the user friendly interface during operation of the FPGA, Labview was used. With this GUI software, it is possible to call C functions if they are included in a shared library, namely a Dynamic Link Library (DLL). The functions within the DLL can execute any routine which calls ZestSC2 functions to configure the FPGA or transfer data. For example, a function could be created to write on the RAM of the FPGA a value x in an address a, where x and a are variables given by the user.

The procedure to generate a DLL library and how to call its functions from LabView is described in National Instruments' documentation "Using External Code in LabVIEW" (preferable the `lvexcode.pdf` file) [lvexcode, 2003]. In Labview, the block that calls a C function contained in a DLL is the *Call Library Function Node* (in Labview's french version is called *Appeler une fonction d'une DLL*). Note that this block can only be called from the block diagram environment and not from the front panel.

C function's definition and declaration to be exported from the DLL must be prefixed by the keyword `_declspec(dllexport)`. For example,

```
_declspec(dllexport) int myfunction(float a, int b);
_declspec(dllexport) int myfunction(float a, int b)
  {...}
```

It must be checked also that the declaration of the C function corresponds to the prototype generated by the *Call Library Function Node* after configuration within LabView.

Steps to generate a DLL differ between versions of Visual Studio software. General steps are the following

- Create an empty DLL project (file → new)
- Add the C file and required libraries (i.e. Zest.h, extcode.h, ZestSC2.lib, SetupAPI.lib) to the project
- In the project settings (right click on project's name, and select settings) change if necessary the following configuration properties:
  - C/C++ → Code generation → Struct member alignment control = 1 Byte
  - C/C++ → Code generation → Run-time library = Multithreaded Debug DLL

- Build the DLL.

The DLL file has to be copied in the same folder as the Labview project. When using DLL calls in Labview, Visual Studio must be also installed on the computer.

3.4. Built applications

The software tools just presented were used to develop utilities which use the FPGA as well as other resources in the ZestSC2 card. These applications are focused on improving the performance of quantum communication and quantum computing experiments. To be completely useful, they need either a system to receive signals from measurement devices (chapter 2) or a medium to drive electro-optical modulators (chapter 4).

3.4.1. Event frequency measurement

A commonly requested task in quantum experiments is to count. How many photons are generated by a light source? How many of them take a certain interferometer path? Have a $|H\rangle$ polarization state? A $\frac{1}{\sqrt{2}}(|s\rangle + |l\rangle)$ time-bin state? Are entangled? Are lost? Due to detector's efficiencies $\eta < 1$ and other limitations, these questions have no direct answers. However, if the experiments are repeated, statistics give a good picture of the behaviour of a particular system.

A more useful statistic than a rough counting of events is the event frequency. A live update of this statistic helps to make alignments in an efficient way. The typical measurement unit used in photon based experiments is counts per second.

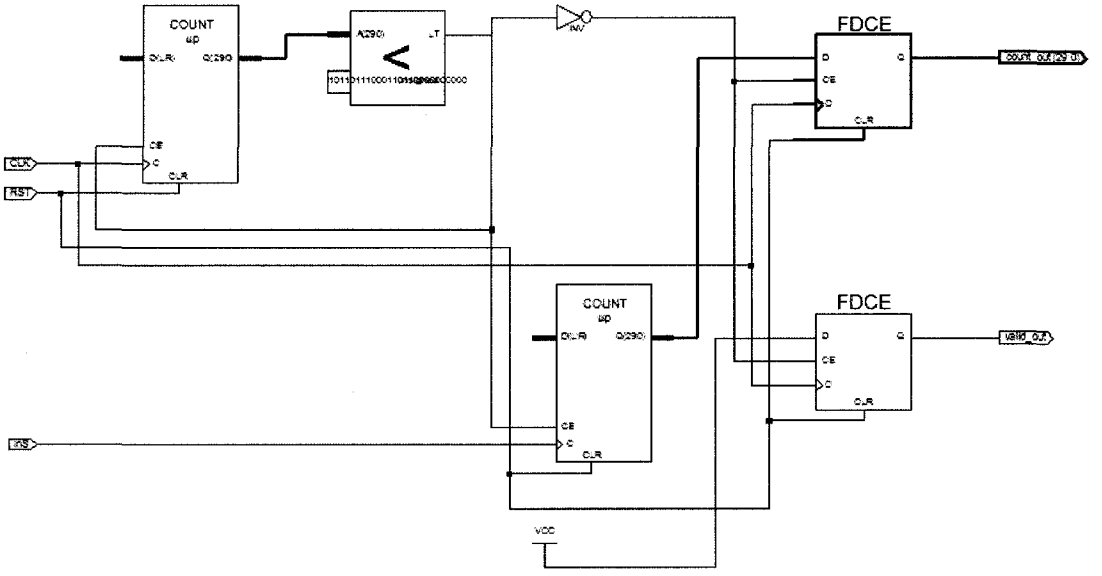


Figure 3.3 RTL of main block for event frequency measurement. Produced with Xilinx ISE. Inputs: clock, reset, 1bit signal. Outputs: integer number of counts, valid data flag.

3.4.1.1. Solution's principle

With the FPGA, it is possible to have blocks counting the rising edges of a signal. Let us use two counters, C_{CLK} and C_{InS}, both with reset and enable options. The value C_{CLK} counts clock cycles that occur since the start of the routine while C_{InS} keeps count of the measured signal rising edges. The clock period T_{CLK} determines the number of system cycles that are required to reach one second. The idea is to use the first one as an enable of the second one, in such a way that C_{InS} works while $C_{\mathrm{CLK}} \leq T_{\mathrm{CLK}}$.

The system saves the C_{InS} value when $C_{\mathrm{CLK}} = T_{\mathrm{CLK}}$. The system then enters in a standby state. Once the registered value is read by the host computer, the reset signal is activated clearing the counters' value and enabling them. The RTL of the developed VHDL code is illustrated in figure 3.3.

The counter value is saved in 4 bytes (32 bits). The computer reads a byte at a time, so

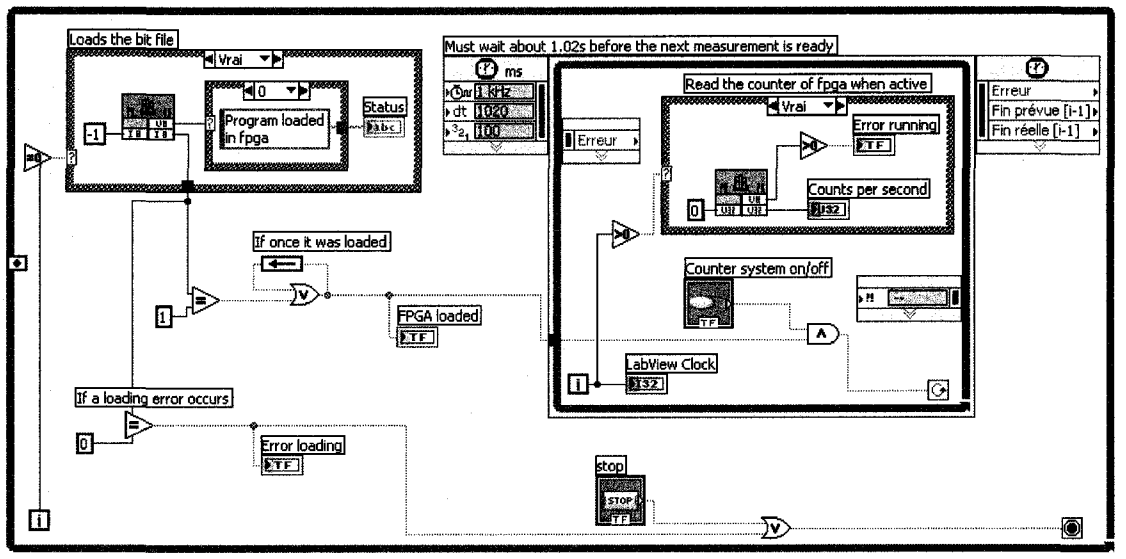


Figure 3.4 LabVIEW block diagram for event frequency measurement routine.

four reading instructions have to be performed. When the fourth one happens, then the system proceeds to the reset state. The computer takes the four values and regenerates the 32bit word containing the *counts per second* measured integer value.

A LabView interface was developed to provide a tool with which provides live results. The particular code was made to measure an event frequency on port IO(2) from FPGA, with $T_{\text{CLK}} = 1/48\text{MHz}$. The LabView routine (figure 3.4) first calls the function `loadtcounter`, which loads the bit-stream file in the FPGA. A loop is then activated periodically each $t = 1.02$ s, where the function `readcounts` is called. This function reads the four bytes and returns the measured event frequency. The execution time is set to $t > 1$ s to permit the FPGA to measure data ($t = 1$ s) and allow the computer to read the four FPGA internal registers. Asking for a result at exactly $t = 1$ s exposes the system to have no measurement available in memory for a read event.

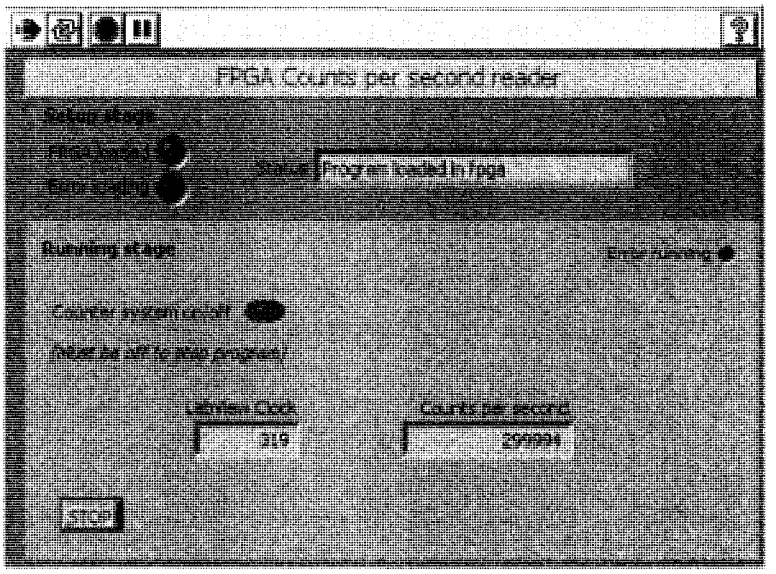


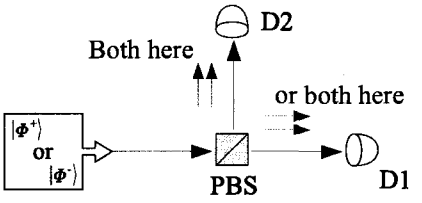
Figure 3.5 LabVIEW front panel for event frequency measurement routine.

3.4.2. Coincidence detection

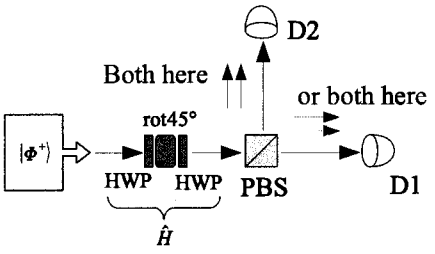
One of the most useful and critical tasks in quantum information experiments is the detection of simultaneous events. When entanglement (section 1.1.1) has to be verified, measurements in particular basis for each qubit contained in the quantum state are applied. Take the following example: suppose that an optical source creates one of the following two photon entangled states in polarization encoding $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|HH\rangle + |VV\rangle)$ or $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|HH\rangle - |VV\rangle)$. It is wanted to distinguish between them.

Measurements of each photon is performed with polarization beam splitter (PBS), where horizontal polarized light goes through, and vertical polarized light gets reflected with a 90° angle. Since both photons are entangled in such a way that they always have the same polarization ($|H\rangle$ or $|V\rangle$) the photons will go to the same detector in the setup of figure 3.6(a). Both photons will be transmited or both will be reflected.

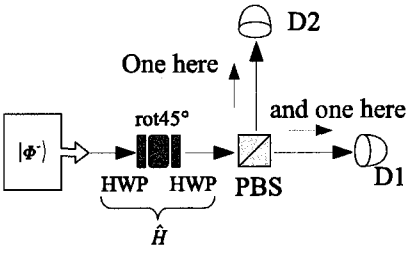
If a Hadamard transformation is applied to each photon prior to the PBS, the states $|\Phi^+\rangle$ and $|\Phi^-\rangle$ become immediately distinguishable. For the state $|\Phi^+\rangle$, applying $\hat{H}$ to each



(a) Effect of a PBS on $|\Phi^+\rangle$ and $|\Phi^-\rangle$. A coincidence never occurs.



(b) Effect of a Walsh-Hadamard gate followed by a PBS on $|\Phi^+\rangle$. A coincidence never occurs.



(c) Effect of a Walsh-Hadamard gate followed by a PBS on $|\Phi^-\rangle$. A coincidence always occurs.

Figure 3.6 Experimental setup to distinguish between $|\Phi^+\rangle$ and $|\Phi^-\rangle$.

photon gives

$$\begin{aligned}
\hat{H}_1 \otimes \hat{H}_2|\Phi^+\rangle &= \hat{H}_1 \otimes \hat{H}_2 \frac{1}{\sqrt{2}}\left(|H_1H_2\rangle + |V_1V_2\rangle\right) && (3.3)\\
&= \frac{1}{\sqrt{8}}\left(|H_1+V_1\rangle|H_2+V_2\rangle + |H_1-V_1\rangle|H_2-V_2\rangle\right) && (3.4)\\
&= \frac{1}{\sqrt{8}}\left(|H_1H_2 + H_1V_2 + V_1H_2 + V_1V_2 + \right. \\
&\quad \left. + H_1H_2 - H_1V_2 - V_1H_2 + V_1V_2\rangle\right) && (3.5)\\
&= \frac{1}{\sqrt{2}}\left(|H_1H_2 + V_1V_2\rangle\right) && (3.6)\\
&= |\Phi^+\rangle && (3.7)
\end{aligned}$$

In other words, $|\Phi^+\rangle$ remains unchanged with a $\hat{H} \otimes \hat{H}$ operation. Now, for the $|\Phi^-\rangle$ state

$$\begin{aligned}
\hat{H}_1 \otimes \hat{H}_2|\Phi^-\rangle &= \hat{H}_1 \otimes \hat{H}_2 \frac{1}{\sqrt{2}}\left(|H_1H_2\rangle - |V_1V_2\rangle\right) && (3.8)\\
&= \frac{1}{\sqrt{8}}\left(|H_1+V_1\rangle|H_2+V_2\rangle - |H_1-V_1\rangle|H_2-V_2\rangle\right) && (3.9)\\
&= \frac{1}{\sqrt{8}}\left(|H_1H_2 + H_1V_2 + V_1H_2 + V_1V_2 + \right. \\
&\quad \left. - H_1H_2 + H_1V_2 + V_1H_2 - V_1V_2\rangle\right) && (3.10)\\
&= \frac{1}{\sqrt{2}}\left(|H_1V_2 + V_1H_2\rangle\right) && (3.11)\\
&= |\Psi^+\rangle && (3.12)
\end{aligned}$$

This quantum state has been changed. When the state $|\Psi^+\rangle$ passes through the PBS, one photon is transmited while the other one is reflected. Which one goes where is irrelevant. In any case, both detectors click at the same time (assuming perfect efficiency).

As a conclusion, by implementing the experiment shown in figures 3.6(b) and 3.6(c) the entangled states $|\Phi^+\rangle$ and $|\Phi^-\rangle$ are easily distinguishable. If no coincidence occurs the produced state is $|\Phi^+\rangle$; if both detectors click simultaneously the state is $|\Phi^-\rangle$.

In practice, the definition of a coincidence must be refined. In theory a coincidence happens when both detectors click within a time interval $\Delta t < \tau_{\text{coh}}$, where τ_{coh} is the coherence time of the source. For our case this time ranges between 1 ps $< \tau_{\text{coh}} <$ 1 ns, corresponding to the coherence times of a Ti:sapphire laser and a Nd:Yag laser. In practice, coincidence is defined as events within a finite time difference Δt. Jitter from detectors and other electronic systems used (i.e. FPGA and conversion circuit) are typically the most significant determining factor for setting Δt. Possible experiment misalignments have also to be taken into account, since a non-zero optical path length difference will also increase Δt.

3.4.2.1. Asynchronous solution's principle

The objective is to make a system capable of determining when two events seem to be in a coincidence. A photon detector first produces a rising edge when it clicks. The electrical pulse duration in a high state is correlated to detector's physical properties rather than the photon's arrival time. Let's say, for photon detectors A and B, that their output signals are D_A and D_B respectively. Define the time when D_A (respectively on D_B) shows a rising edge as t_{r-A} (respectively on t_{r-B}).

The desired system sends a rising edge if $|t_{r-B} - t_{r-A}| < t_{\text{wait}}$, where t_{wait} is the maximum waiting time between a click in detector A (respectively on B), and in detector B (respectively on A). The solution architecture is based on the block shown in figure 3.7. The detector signals D_A and D_B drive the clock signals of a couple of D-type flip-flops with reset signals.

When D_A presents a rising edge, its corresponding flip-flop saves a 1. It is erased after the time given by the feedback delay. Therefore, this generates a short pulse after a click is received. The same process occurs with signal D_B.

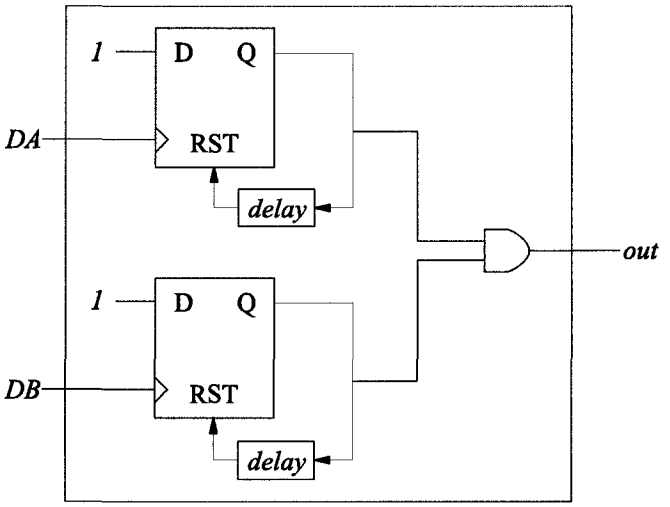


Figure 3.7 Architecture for asynchronous coincidence detection.

If these two pulses happen at the same time, or one starts with respect to the other in a time under the delay time, the AND-gate at the end of the architecture creates a pulse per detected coincidence.

3.4.2.2. Synchronous solution's principle

Here an alternative solution is presented. The advantage of a synchronous solution is the independence of adjusting delays within the architecture block; the disadvantage is its dependence on the system's clock.

As in the asynchronous solution, the system sends a rising edge if $|t_{r-B} - t_{r-A}| < t_{\text{wait}}$. However, these three times (t_{r-B}, t_{r-A} and t_{wait}) are going to be rounded to the next clock rising edge; therefore the system is synchronous.

The logic works as follows: for a 200 MHz clock (provided by a DCM), each 5 ns the system reads the state on inputs D_A and D_B and compares it with its pervious value (5 ns ago). Let us start with the state where both have not a detection; $D_A = D_B = 0$. If D_A (or D_B) changes to 1, a timer t_{timer} starts running. This timer runs while $t_{\text{timer}} \leq t_{\text{wait}}$,

Table 3.3 Behaviour of coincidence detection with synchronous approach. States are represented by the logic levels of each detector (00 means both detectors show low levels).

Previous state	New state	Condition on timer	Result
00	11	no condition	coincidence
01	11	if $t_{\text{timer}} \leq t_{\text{wait}}$	coincidence
10	11	if $t_{\text{timer}} \leq t_{\text{wait}}$	coincidence
00	01	activate timer	
00	10	activate timer	
XX	00	no condition	reset system

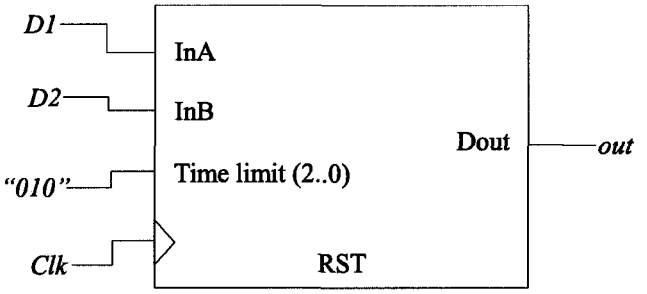


Figure 3.8 Architecture block for synchronous coincidence detection.

giving a chance to D_B (or D_A) to change to 1. If both detector signals become 1 before their timer runs out, a coincidence is counted.

If the signals change from both being 0 to both being 1 within a clock period, the system immediately recognizes this situation as a coincidence. This means that the timing resolution for coincidence counting is given by the system's clock. Note that this procedure starts only from both detectors being in 0, and that it works independently of which of the detectors clicks first.

The inputs for this architecture block are a reset, a clock, the two inputs from the detectors and the value of t_{wait}, which corresponds to $T_{\text{CLK}} \times$ 'time limit' in figure 3.8.

3.4.3. Time stamping

To collect statistics from photon arrivals, coincidences and waiting times it is useful to know not only which event happens, but when it does happen. Since the FPGA has a clock, a chronometer can be set. Each time that an event occurs in a signal (rising edge or falling edge), the change and the moment when it happened can be recorded.

There are two main limitations for a time stamping utility. The first is the time resolution: if the FPGA clock is too slow, it may miss pulses shorter than the period of the clock T_{CLK} . For example, the sync out signal from the Mira 900 Laser of 76 MHz cannot be time stamped with the default Spartan-3 $f_{\text{CLK}} = 48$ MHz included in the ZestSC2 card. The second is memory: saving for a long time may overflow the memory capacity. Also when the recorded elapsed time increases, its value in clock ticks requires a higher amount of bits in memory.

To obtain a better performance in terms of recording speed, a DCM (Digital Clock Manager) from the FPGA is used. This results in a system operating with two clocks at the same time, since the original frequency $f_{\text{CLK}} = 48$ MHz has to be employed for the already configured communication with the host computer.

The DCM is synthesized with the *Architecture Wizard* tool from Xilinx, which creates a VHDL code that can be inserted in the project. The maximum frequency that can be obtained is $f_{\text{CLKFX}} = 208$ MHz, which corresponds to equation (3.2) with $f_{\text{CLKIN}} = 48$ MHz, $M = 13$ and $D = 3$. However tests were made with a synthesized frequency $f_{\text{CLKFX}} = 200$ MHz ($M = 25$ and $D = 6$).

To interconnect the systems that save or transmit data at different rates, a dual clock FIFO was used. This is a FIFO where writing operations work at a different rate from reading operations. In this particular case the writing clock frequency exceeds the reading one

$f_{\text{write}} > f_{\text{read}}$. Care has to be taken since the dual clock FIFO may get overflowed. Flag treatment can help to detect this situation and avoid misinterpretations of information when an overflow happens. The dual clock FIFO is synthesized with the help of the *CORE Generator* tool from Xilinx, more precisely the *FIFO Generator*. Independent clocks option is selected. Full, empty and valid flags are activated.

To reconstruct all the signal information in small packages of information, each time that a rising or falling edge occurs in a tracked signal, the whole status of the tracked signals is saved in conjunction with the corresponding time stamp. To avoid large time stamps (in terms of bits) whenever the timer overflows, the status of the tracked signals with the time stamp is also saved. In the case of a b-bits time stamp t_s, once $t_s = 2^b - 1$ the status of the n tracked signals $s[0 \ldots n]$ is saved. This prevents having long bit chains for timing information. In the other hand, if the bit chain is made too small the FIFO may rapidly overflow.

The read data from the dual clock FIFO is then saved in the FPGA's main FIFO. The host computer reads this from the FPGA using the C function `ZestSC2ReadData`. Memory has to be previously allocated to receive the output of this function. The transferred data from the FPGA to the computer via streaming is then saved in a plain text file. The C code also reconstructs the actual time value in FPGA ticks; this means that each time the b-bits time stamp information is $t_s = 11 \ldots 11 = 2^b - 1$, a local variable `myclock` increases by 2^b. For any event registered, the time saved in the plain text file is then `myclock` $+ t_s$, corresponding to the time elapsed since the start of the routine measured in FPGA ticks.

It is good to remember that the time resolution of the system shifts or even erases timing information. Also all amplitude information is lost; this is intended for digital signal recording.

3.4.4. Pseudo-random number generator

In quantum communication protocols, random number generators (RNGs) are frequently required, as explained in section 1.2. Physical ways to generate random numbers such as flipping a coin or throwing a dice exist. It would be interesting if an electronic system like an FPGA could make a RNG. This would allow us to have all what is necessary to control a quantum communication protocol that requires random numbers in one independent system.

Electronic systems are generally incapable of producing a RNG, since their numbers are generated from an algorithm that can be repeated, and therefore the output numbers are predictable if the algorithm is known. What electronic systems implement is known as pseudo-random number generator (PRNG). One particular kind of PRNG is a linear feedback shift register (LFSR). It is widely used because it requires few computational resources.

An LFSR is a sequence of n-bit registers that are shifted in a synchronous way. To put it in mathematical terms, lets say that the ith bit is given by the polynomial x^{i-1} multiplied by its value (0 or 1); so the first bit corresponds to the polynomial order $x^0 = 1$, the second to $x^1 = x$, the third to x^2 and so on. A shift corresponds to multiply the polynomial by x. For example if $A = 0011010 = a_6 a_5 \ldots a_0$ is the 7-bit number to be shifted, its polynomial is

$$A = a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0 \cdot 1 \quad (3.13)$$

$$A = 0 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 0 \cdot 1 \quad (3.14)$$

$$A = x^4 + x^3 + x \quad (3.15)$$

Proceeding to the shift operation, forgetting that the most significant bit (first digit) goes

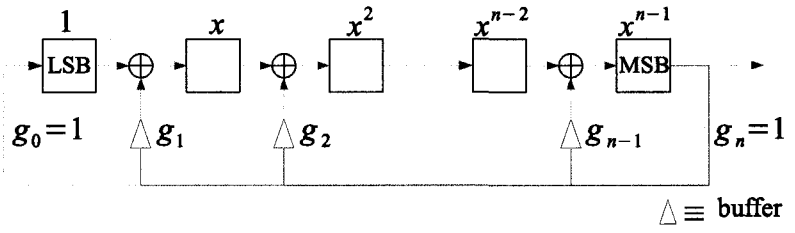


Figure 3.9 LFSR Galois general scheme. Each buffer allows the signal pass by when $g_i = 1$, and therefore apply the corresponding xor-gate.

back in the chain to become the least significant bit (last digit), the new polynomial xA is

$$xA = x^5 + x^4 + x^2 \tag{3.16}$$

which corresponds to $xA = 0110100$. The principle of an LFSR is to modify at least one of the shifted bits replacing it (or them) by the result of an exclusive-or ('xor', represented by $\oplus$) operation between two established bits. There are two types of LFSR: Fibonacci and Galois. Here the LFSR Galois is described and implemented.

Some bits are chosen to be potentially changed when a shift occurs. These chosen bits are changed if they are equal to the most significant bit (MSB), the one that is becoming the least significant one (LSB) after shifting. If they are different, they remain unchanged. This is nothing but a 2-input xor between each of them and the returning bit. This can be seen in figure 3.9. Each coefficient g_i indicates if the xor for the ith-bit is applied or not with the n-bit. Notice that $g_0 = 1$ and $g_n = 1$ always.

The characteristic polynomial of an LFSR is given by

$$G(x) = g_n x^n + g_{n-1} x^{n-1} + \ldots + g_1 x + g_0 \tag{3.17}$$

For example, the characteristic polynomial of the LFSR shown in figure 3.10 is $G(x) = x^3 + x + 1$, since the $g_2 = 0$ cable is the only one deactivated. There exists another way

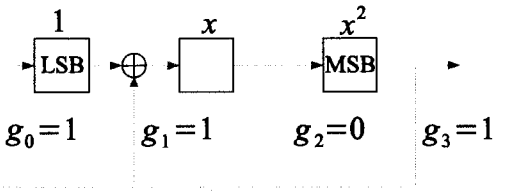


Figure 3.10 LFSR Galois example with characteristic polyomial $G(x) = x^3 + x + 1$.

to represent a characteristic polynomial in a more compact way, called the *feedback taps*. It corresponds to the list of activated indexes $g_i = 1$, in decreasing order. The first index indicates the length of the LFSR n, being always $g_n = 1$. The $i = 0$ case is omitted since by definition $g_0 = 1$. Continuing with the example of figure 3.10, its feedback tap representation is $[3, 1]_g$, where the g subindex indicates that the LFSR is a Galois type.

An LFSR is then a cyclic series of bits that appear to be random. It is cyclic since any value is followed by a single possible value, and the number of values that can be made with a n-bit string is 2^n. Notice however that there is one value that doesn't work for an LFSR for producing pseudo-random numbers: zero. If a sequence of only zeroes starts the sequence, this value remains, since $0 \oplus 0 = 0$.

In this order of ideas, the best that an LFSR can do is a periodic sequence of numbers with period $2^n - 1$. A sequence that has this particular period is called a *maximum sequence* or *m-sequence*. They represent the closest result to a RNG. When one of the word's bit is taken as the output of the system b_{out}, it doesn't represent a random series since the probabilities of obtaining $b_{\text{out}} = 0$ or $b_{\text{out}} = 1$ are not equal.

$$P(b_{\text{out}} = 0) = \frac{1}{2^n - 1}\left(\frac{2^n}{2} - 1\right) \tag{3.18}$$

$$P(b_{\text{out}} = 1) = \frac{1}{2^n - 1}\left(\frac{2^n}{2}\right) \tag{3.19}$$

Notice that increasing n, makes both probabilities closer to $\frac{1}{2}$. Maximal and near-

maximal LFSRs can be theoretically determined [Clark and Weng, 1994].

For this project a $[16, 13, 12, 7]_g$ was implemented. In general, an LFSR Galois with four taps $[A, B, C, D]_g$, can be implemented by the following VHDL instruction

```
lfsr <= lfsr(A-2 downto B) & (lfsr(A-1) xor lfsr(B-1))
          & lfsr(B-2 downto C) &(lfsr(A-1) xor lfsr(C-1))
          & lfsr(C-2 downto D) & (lfsr(A-1) xor lfsr(D-1))
          & lfsr(D-2 downto 0) & (lfsr(A-1) xor lfsr(0));
```

In the case that $A = B + 1$, $B = C + 1$ or $C = D + 1$, the corresponding `downto` sequence must be eliminated. Therefore, for $[16, 13, 12, 7]_g$ the sequence "`& lfsrout(B-2 downto C)`" has to be removed, since $B = 13$ and $C = 12$.

3.4.5. Periodic signal generator

In time-bin encoding system for quantum communication and computing it is useful to have periodic digital signals at precise frequencies. These signals are used to drive amplitude and phase modulators within a time-bin based set-up.

The frequency of the signals corresponds to the one given by interferometers used in the experiment. Usually the interferometers are in fact designed to work at a given frequency imposed by the control system. They may be used in state preparation stages as well as measurement stages.

It is desirable that the electronic system generates these periodic signals by itself. In this project, the FPGA was used to generate 45 MHz and 90 MHz periodic signals that were required for a particular custom-made interferometer.

Table 3.4 Encoded output value for periodic signal generator application.

Code value	Signal
0x00	logic '0' (GND)
0x01	logic '1' (VCCO)
0x02	45 MHz clock signal
0x03	90 MHz clock signal
0x04	45 MHz clock signal with a 180° phase

The periodic signals were synthesized with a DCM from the FPGA. A program was made routing the DCM produced signals and making them available in 5 pins of the IO2 port of the FPGA. The VHDL code description multiplexes one of five possible values for a pin: 0 , 1, a 45 MHz periodic signal with zero phase or 180° phase, or a 90 MHz periodic signal.

To drive them from the computer, in VHDL the selection of the output values IO2[4 . . . 0] each depended on a registered value at the addresses 0x200A + i, where $0 \leq i \leq 4$ indicates the IO2 corresponding port. So, the computer host program writes in the desired position the wanted encoded output value, as shown in table 3.4.

A user interface in LabView was developed. It calls a load function (`loadclocksGB`) to configure the FPGA at the beginning of the routine, and it calls the write instruction `writereg` when a change in the roll-down menus value occur, at the front panel (shown in figure 3.12). Each chosen value calls the same function but sends different parameters.

The particular case of a change in IO4 is illustrated in figure 3.11, where the address variable value is 0x0A + 4 = 14, since the C code adds to this value 0x2000, obtaining the address 0x200E. The encoded value sent to the function comes from the drop-down selection.

The needed output to drive the phase modulator is 5 V amplitude. The ZestSC2 card is not capable of generating this kind of signal. The maximum output amplitude produced

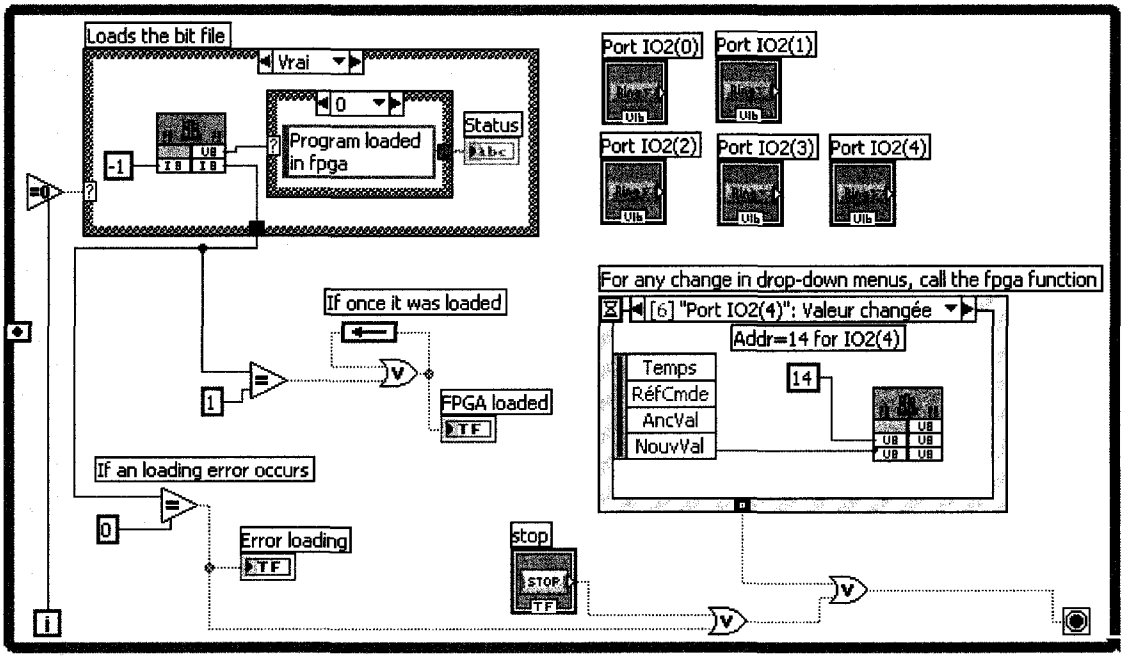


Figure 3.11 LabVIEW block diagram for periodic signal generator routine.

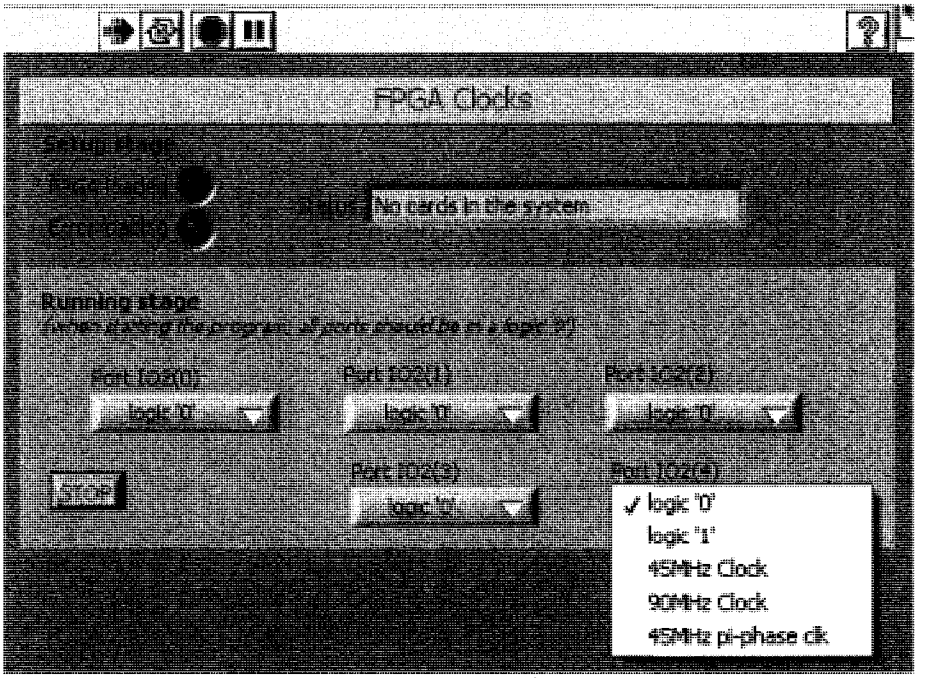


Figure 3.12 LabVIEW front panel for periodic signal generator routine.

by the FPGA is 3.3 V ideally, but since the current output of the FPGA is restricted, the actual value achieved with a single output port was about 1.8 V. This is due to the load of the electro-optical modulators is 50 Ω.

A temporary solution is to apply the same signal in several output ports, and putting them in parallel to drive as much current as possible from the ZestSC2 card. Amplitudes of 2.9 V were thus obtained. Nevertheless the shape of the periodic signal becomes distorted when adding the signals in this way. This happens because the signals do not arrive at the same moment to their intersection point. This method is not suggested. The right way to obtain periodic signals with higher amplitudes is addressed in chapter 4, where a circuit is designed for this purpose.

3.5. Chapter summary

The properties and operation of the used FPGA and the card that contains it are introduced. Several applications were developed. These applications respond to particular necessities of quantum information experiments.

Event frequency measurement (section 3.4.1), coincidence detection (section 3.4.2) and time stamping (section 3.4.3) applications acquire data from external devices. This acquisition can be done from different types of analog signals, converted to digital signals by the input circuit described in chapter 2.

Pseudo-random number generator routine (section 3.4.4), as the periodic signal generator one (section 3.4.5) are intended to give an output signal, which can be used to drive electro-optical modulators. These signals however require extra treatment in order to drive modulators; this is the subject of chapter 4.

CHAPTER 4

OUTPUTS AND CONTROL

In previous chapters, the processes of photon detection, reception and processing have been covered. The generation of data and output signals that drive outer electronic systems for control issues is the last link in the chain. Some output characteristics have been already presented in section 3.4.

Among the problems to solve at the output stage is the compatibility of signals, a situation analogous to the one present at the inputs of the card. Output signals serve several purposes, and each of them may require a particular signal treatment. For this project, it is of particular interest to drive electro-optic modulators, as explained in section 4.1. Among them, amplitude and phase modulators are of interest. Their working principle is presented. To drive them an amplification system is proposed in section 4.2. To drive them, a custom circuit was designed and fabricated, which is presented in section 4.3.

4.1. Electro-optic modulators

Variation of refractive index can be induced in some special crystals by applying an electric field. This electro-optic phenomenon is known as the *Pockels effect* when the dependence of the refractive index is linear with the electric field and *Kerr effect* when it is quadratic. For modulation purposes, a linear dependence is more practical, and therefore the Pockels effect is preferred. For this case the refractive index goes as

$$n(E) \approx n_0 - \frac{{n_0}^3 r E}{2} \tag{4.1}$$

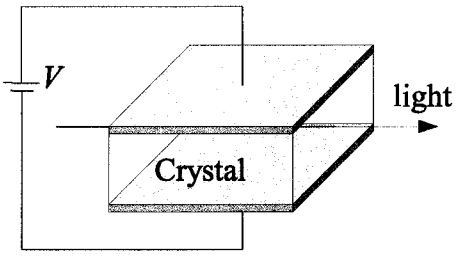


Figure 4.1 Phase modulator made with a non-linear crystal inside a capacitor.

where $n_0 = n(0)$, and r is known as the electro-optic coefficient. Constant values typically range from 10^{-12} to 10^{-10}m/V. A *Pockels cell* is made of a non-linear crystal where metallic parallel plates are placed (see figure 4.1). This forms a capacitor in the crystal, which provides a constant electric field in it $E = V/d$ where d is the distance between the conductor plates. A variation in the voltage has as consequence a linear change in the refractive indices of such a crystal.

Some of the non-linear crystals that present this effect are: potassium di-deuterium phosphate (KD*P = DKDP), potassium titanyl phosphate (KTP), beta-barium borate (BBO), lithium niobate ($LiNbO_3$), lithium tantalate ($LiTaO_3$), ammonium dihydrogen phosphate $NH_4H_2PO_4$ (ADP) and cadmium telluride CdTe [Saleh and Teich, 1991].

4.1.1. Phase modulators

The first application of the Pockels effect is a phase modulator. When light passes by this cell with V applied, the velocity of light inside the crystal is changed, therefore its phase is modified. Furthermore, the total accumulated phase in the crystal is

$$\phi = n(E)\frac{2\pi}{\lambda_0}L \tag{4.2}$$

if light passes by the crystal a length L and its wavelength in free space is λ_0. Using equation (4.1)

$$\phi = \left(n_0 - \frac{{n_0}^3 r}{2} E\right) \frac{2\pi}{\lambda_0} L \tag{4.3}$$

$$\phi = \phi_0 - \pi \frac{{n_0}^3 r E L}{\lambda_0} \tag{4.4}$$

$$\phi = \phi_0 + \Delta\phi \tag{4.5}$$

There exists a particular voltage for which the additional phase $|\Delta\phi|$ is π. This voltage is known as V_π or half wave voltage. This value is an important specification that is usually given by optical phase modulator manufacturers. The electric field required for a π phase is $\lambda_0/({n_0}^3 r L)$, and therefore

$$V_\pi = \frac{d\lambda_0}{{n_0}^3 r L} \tag{4.6}$$

Most phase modulators are polarization dependent, which means that only a particular polarization maximizes the phase shift; this requires incident light to be aligned with a preferred transmission axis. Temperature fluctuations also change the refractive index of the material and subsequently modify the phase applied at a given voltage. This means that V_π depends on temperature.

The operation frequency of phase modulators can go as high as 40 GHz and V_π values can go as low as a few volts. These limit values are achieved with micro-fabricated waveguides made of $LiNbO_3$ with on-chip electric contacts. In this type of fabrication, an electrode impedance of 50Ω is usual.

4.1.2. Amplitude modulators

Phase modulators can be used to make amplitude modulators. The idea is to use a Mach-Zender interferometer and control the phase applied in one of the interferometer arms. By changing this phase, constructive or destructive interference occurs at the output junction, producing the amplitude modulation.

The characteristic voltages in an amplitude modulator are those that determine a constructive interference where no phase difference exists between arms, and a destructive interference where the phase difference between paths is π. For constructive interference an amplitude 1 is defined and its voltage is named V_1; for destructive interference an amplitude 0 is defined and its voltage is named V_0. Notice that when using a phase modulator for making an amplitude modulator $V_\pi = V_1 - V_0$.

4.2. Electric signal amplification

Phase and amplitude modulators are driven with periodic signals. A method for generating these signals with an FPGA was developed and described in section 3.4.5. An arbitrary phase modulation ϕ is obtained with a voltage $V_\phi = \phi V_\pi / \pi$. The obtention of the required voltage typically requires amplification of the FPGA output signal.

An operational amplifier seems a logical solution. This is not the case, however, because operational amplifiers don't offer a sufficiently good frequency. Currently, what are known as "very high speed operational amplifiers" have "reasonable performance up to 50 MHz" [LT1226, 1992, p.6]. A correct solution for this case, in the hundredths of MHz regime, is to use transistors as switches.

In quantum communication protocols implemented over time-bin encoding, discrete pre-

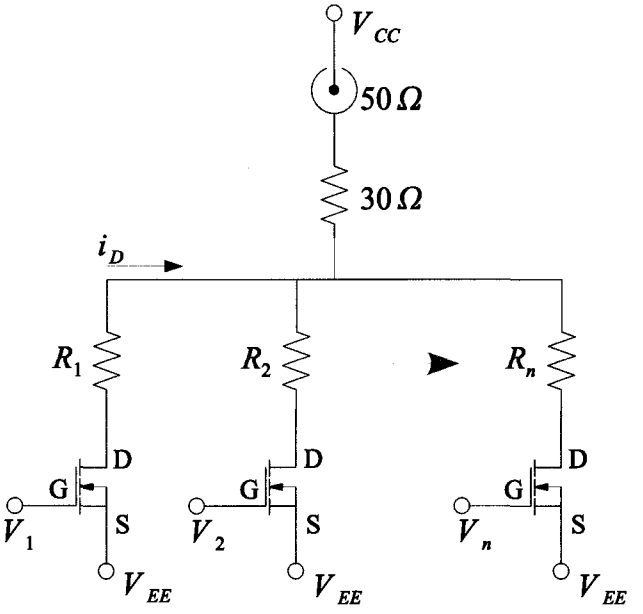


Figure 4.2 Phase modulator driver circuit scheme, with n possible phase values.

determined values of phase modulation are used. To prepare states or to make change in between the $|0\rangle$, $|1\rangle$ and $\frac{1}{\sqrt{2}}|0+1\rangle$, $\frac{1}{\sqrt{2}}|0-1\rangle$ basis, typical phase values required are $\pi/4$ multiples $(0°, 45°, 90°, 270°, \ldots)$. For some protocols other values may be wanted, as in [Berlín et al., 2008a] where a phase of $36°$ is also required. In the case of amplitude modulators, they are used as optical switches, either blocking or transmitting photons. Therefore V_0 and V_1 are the only values of interest.

The chosen scheme to obtain several drive levels is presented in figure 4.2. When the gate source voltage V_{GS} of the transistor passes its threshold level ($V_{\text{GS}} > V_{\text{th}}$), a drain current i_{D} flows. This current provides a voltage in the connector of $V_{\text{out}} = 50\ \Omega \cdot i_{\text{D}}$, since the phase (amplitude) modulator input has $50\ \Omega$ impedance. The system is operated in such a way that only one transistor is 'on' per SMA output connector.

The i_{D} current for a single conducting transistor is

$$i_D = \frac{V_{\text{CC}} - V_{\text{EE}}}{50\ \Omega + 30\ \Omega + R_j + R_{\text{DL}}} \tag{4.7}$$

where R_j is the resistance placed as load in the jth transistor, R_{DL} is the inner resistance of the transistor from drain to load and the 50 Ω corresponds to the impedance of the phase (amplitude) modulator. The V_{EE} choice is given by the threshold voltage V_{th} and the FPGA's output voltage when a logic '1' occurs (let's say $V_{FPGA} = 1.8$ V),

$$V_{FPGA} - V_{EE} > V_{th} \tag{4.8}$$

Since the possible values are determined by the resistances R_i, the phase shift values (or the amplitude modulation values) are fix. The range of voltages is discrete, with n choices.

4.3. The output circuit

Most of the general characteristics of the designed PCB for controlling the electro-optic modulators are the same to those of the input's PCB, from section 2.3. Modeling is performed in Eagle 5.2.0. It is designed to interface with the ZestSC2 card presented in chapter 3.

The transistor selected to this application is an RF power transistor PD57002-E from STMicroelectronics. It is designed to work at frequencies up to 1 GHz. Its maximum V_{DS} is 65 V, where its typical operation is at 28 V. Its maximum drain current i_D is 0.25 A. Its threshold voltage $V_{th} = 3.0$ V.

4.3.1. PCB schematic

The supply voltages of the designed PCB are $V_{CC} = 10$ V and $V_{EE} = -2$ V. When a logic '0' come in the gate pin of the transistor ($V_G = 0$ V), the transistor is off since

$$V_{GS} = 0 - (-2\text{ V}) = 2\text{ V} < 3\text{ V} = V_{th} \tag{4.9}$$

For the logic '1' case, $V_G = 1.8$ V (depending on configuration of ZestSC2 card could be 1.8 V, 2.5 V or 3.3 V) the transistor is on

$$V_{GS} = 1.8\text{ V} - (-2\text{ V}) = 3.8\text{ V} > 3\text{ V} = V_{th} \tag{4.10}$$

Supply voltages V_{CC}, GND and V_{EE} are provided via the connector J1. The inputs for this circuit are provided by a 32x2 pin header connector fitting directly in the J9 or J10 FPGA header of the ZestSC2 card. The outputs of the circuit are SMA connectors for connection with the phase (amplitude) modulators over high frequency coaxial cables.

In order to obtain the desired precision of each resistance value a pair of parallel resistors are placed where one value is just above the desired one and the second value can be 2 or 3 orders of magnitude higher, instead of a single resistor. This is necessary because actual resistor values differ from the advertised value. High precision resistors may help, but still do not provide the required precision. For example, instead of looking for a 30 Ω resistance, is preferable to use $30.9\ \Omega||1.02\ \text{k}\Omega \approx 30.0\ \Omega$. Values of R_i to obtain some phase modulations φ of interest are shown in table 4.1.

For the case where the only two resistance connected to the drain of a transistor are 30 Ω

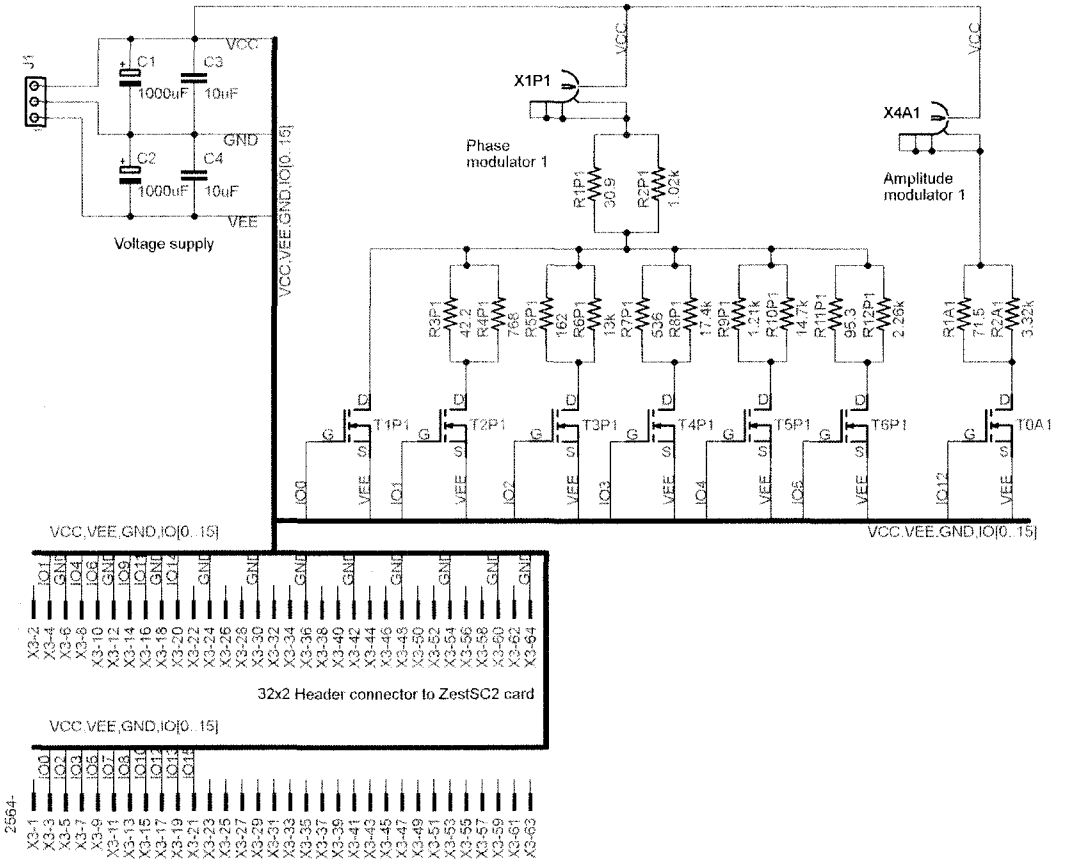


Figure 4.3 Schematic of designed output PCB. The treatment circuit for a single phase modulator and a single amplitude modulator is here depicted. The actual PCB drives four electro-optical modulators (2 of each type).

Table 4.1 Resistance values to obtain some phase modulation values using the scheme from figure 4.2, if $V_\pi = 5.0$ V. R_{low} and R_{high} have commercial values.

$\mathbf{V_{out}}$	φ		$\mathbf{R_i}$ ($\mathbf{R_{low}}$ $\mathbf{\|\|}$ $\mathbf{R_{high}}$)			$\mathbf{R_{low}}$	$\mathbf{R_{high}}$
7.5 V	270°	$\frac{3\pi}{2}$	30 Ω − 30 Ω	=	0 Ω		
5.0 V	180°	π	70 Ω − 30 Ω	=	40 Ω	42.2 Ω	768 Ω
3.5 V	126°	$\frac{7\pi}{10}$	121.4 Ω − 30 Ω	=	91.4 Ω	95.3 Ω	2.26 kΩ
2.5 V	90°	$\frac{\pi}{2}$	190 Ω − 30 Ω	=	160 Ω	162 Ω	13 kΩ
1.0 V	36°	$\frac{\pi}{5}$	550 Ω − 30 Ω	=	520 Ω	536 Ω	17.4 kΩ
0.5 V	18°	$\frac{\pi}{10}$	1148 Ω − 30 Ω	=	1118 Ω	1.21 kΩ	14.7 kΩ
0 V	0°	0	All signals off				

and the modulator's 50 Ω, the drain's current for a 12 V supply ($V_{CC} - V_{EE} = 12$ V) is

$$i_D = \frac{V_{CC} - V_{EE}}{50\ \Omega + 30\ \Omega} = \frac{12\ \text{V}}{80\ \Omega} = 0.15\ \text{A} \tag{4.11}$$

which is less than the transistor maximum of 0.25 A. Notice that selecting a higher ($V_{CC} - V_{EE}$) voltage could draw a higher drain current and damage a transistor.

Each resistor consumes power $P = IV$ for each resistor. Taking the worst case scenarios, it can be determined that the required power ratings for the resistors are: 1 W for the 30 Ω resistors, 0.5 W for the low valued resistors in each couple of resistors and 0.25 W for the high valued ones.

As in the input circuit PCB (section 2.3.1), a 1000 μF||10 μF bypass capacitor is placed next to each supply voltage. No extra capacitors are required to drive signals.

The PCB is designed to drive two phase modulators and two amplitude modulators. Each phase modulator connector has 6 possible values, and the amplitude modulator connector only a single possible value. In table 4.2 is the whole list of required elements for this PCB.

4.3.2. PCB layout

The layout of the PCB required to model the PowerSO-10RF package of the PD57002-E transistor. The specifications are given in an application note of the component producer [AN1294, 2001]. This model requires via holes for proper power dissipation which are connected to the lower voltage layer, i.e. V_{EE}.

As in the input circuit PCB, a 4 layer model was used, where the top and bottom layers are for routing purposes. The second layer is dedicated to GND and the third layer in

Table 4.2 Components in the output circuit PCB.

Device	**Quantity**
Connectors	
SMA PCB horizontal connector	4
Connector header vertical, 3 positions. 0.100in	1
Pin header 32x2. 0.100in	1
Integrated circuits	
PD57002-E transistor	14
Resistors and capacitances	
10 μF tantalum capacitance SMD 1206	2
1000 μF electrolitic capacitance	2
30.9 Ω resistance, 1 W SMD 2512	4
1.02 kΩ resistance, $1/4$ W SMD 1206	4
Low valued resistances, $1/2$ W SMD 2010	2 each
High valued resistances, $1/4$ W SMD 1206	2 each

this case corresponds to V_{EE}.

Routing was manually made for all the signals coming from the FPGA all the way to the SMA connectors, passing through the transistors and resistances. Manual routing is done to ensure signals arriving at the same point passing by routes of the same length; failing to do so may generate overlapping in high frequency digital signals (MHz) and therefore an incorrect output value.

Since phase and amplitude modulators have a 50 Ω impedance input, all the routing was designed to have the same 50 Ω impedance. The width of the microstrip traces is $W = 9.234$ mil$(0.2345$ mm$)$, since the dimensions and characteristics are the same as in the input circuit case (section 2.3.2). The V_{EE} paths have $W = 50$ mil and other regular paths have $W = 10$ mil.

The connections layout for this circuit is given in the appendix I.2.

4.4. Chapter summary

Electro-optical phase and amplitude modulators are driven by periodic signals with amplitudes that determine the amount of modulation applied. Since an FPGA offers only two levels of amplitude as output, corresponding to logic 1 and 0, a circuit for driving these modulators using signals from an FPGA was proposed.

The designed circuit, intended to work with the ZestSC2 card (section 3.2), has four outputs: two of them are capable of applying 6 different voltage values (per channel) and the other two have only one possible voltage value. They are intended to drive phase modulators, where phase changes are frequent, and amplitude modulators, where they are typically used as an optical switch.

This circuit can be placed in two ports of the ZestSC2 card, offering the possibility to drive up to eight electro-optical modulators: four phase and four amplitude modulators.

CHAPTER 5

RESULTS AND ANALYSIS

5.1. Input circuit

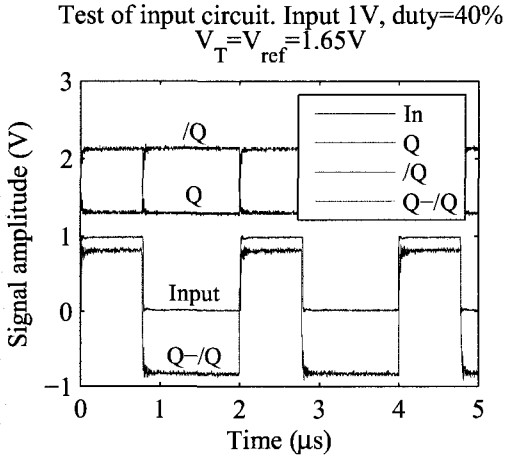
The circuit designed for electric signal standardization, described in section 2.2 was tested. Input signals were generated by using a digital delay/pulse generator (model DG535 from Stanford research systems). Supply voltage for the PCB is set at $V_{CC} = 3.3$ V.

For a 1 V amplitude signal used as input, and a duty cycle about 40%, the results are shown in figure 5.1(a). Here $V_T = V_{\text{ref}}$ (shunting them) are externally set at $V_{CC}/2 = 1.65V$. The signals Q and $/Q$ correspond to the positive and negative output pin, measured with respect to the common ground plane, and with 50Ω impedance at the oscilloscope. The resulting differential signal $Q - /Q$ is also depicted. Notice that the DC level of this signal is almost zero, since its high and low levels are ± 800 mV.

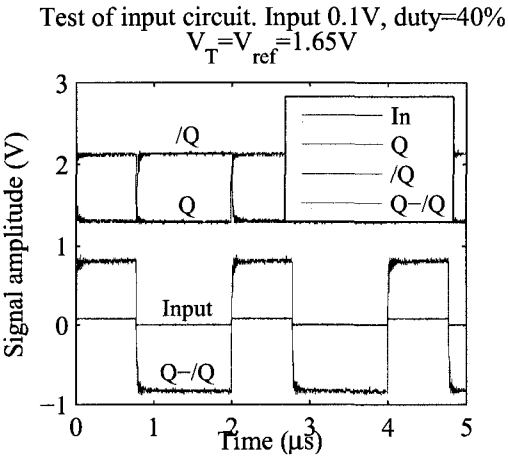
The input signal is then reduced to 0.1 V. The result is shown in figure 5.1(b). The Micrel component stills recognize this weak amplitude signal.

Then the duty cycle is modified. A situation of reduced duty cycle (4%) is tested. The case where $V_T = V_{\text{ref}}$ is in figure 5.2(a). The high and low levels are not any more ± 800 mV. In fact, the high level remains 800 mV, but the low level increases to -100 mV.

A similar situation is presented when testing a high duty cycle case (96%), as shown in figure 5.3(a), where the roles of high and low output levels are inversed; 100 mV for

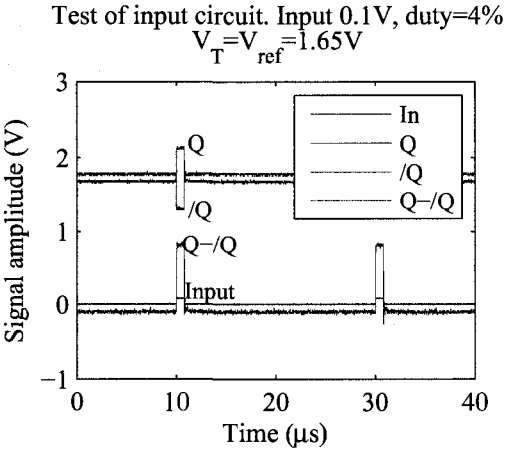


(a) Input of 1 V amplitude. Differential output of ±800 mV.

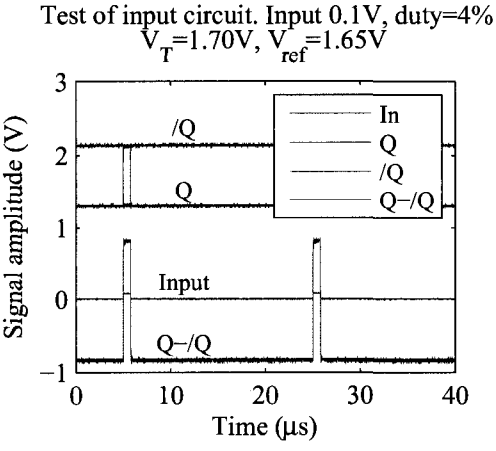


(b) Input of 100 mV amplitude. Differential output of ±800 mV.

Figure 5.1 Test results of designed input circuit, with 40% duty cycle input. $V_T = V_{\text{ref}} = V_{CC}/2$.

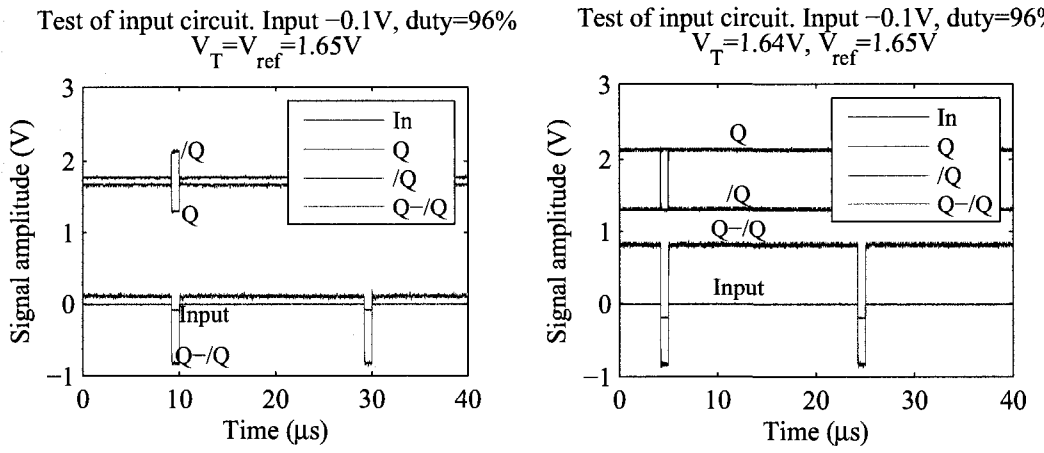


(a) $V_T = V_{\text{ref}}$. The differential output has a wrong low level.



(b) V_T just above V_{ref}. Differential output of ±800 mV.

Figure 5.2 Test results of designed input circuit, with 4% duty cycle 100 mV amplitude input.



(a) $V_T = V_{\text{ref}}$. The differential output has a wrong high level.

(b) V_T just under V_{ref}. Differential output of ±800 mV.

Figure 5.3 Test results of designed input circuit, with 96% duty cycle 100 mV amplitude input.

high level, and −800 mV for low level.

These situations have to be corrected in order to be read by the FPGA correctly. This is adjusted by modifying the V_T value, letting $V_{\text{ref}} = 1.65V$. In the 4% duty cycle case, V_T is increased slightly to 1.70 V. Result is shown in figure 5.2(b). For the 96% duty cycle case, V_T is reduced slightly to 1.64 V, to obtain the results in figure 5.3(b). Corrected ±800 mV differential signals are obtained.

5.1.1. Analysis

When using the input circuit for signals with a duty cycle close to 50%, it can be operated by shunting V_T and V_{ref} at a $V_{CC}/2$ level. This offers a good differential signal as an output, which can be read by the FPGA.

For narrow or wide pulses the optimal operation parameters are setting $V_{\text{ref}} = V_{CC}/2$

Table 5.1 Event frequency measurement application tests. A variable frequency TTL signal generator is used as source.

Real frequency (Hz)	700000	660000	620000	580000	540000	500000
Measured events (cps)	699986	659987	619988	579988	539989	499990
Real frequency (Hz)	460000	420000	380000	340000	300000	260000
Measured events (cps)	459991	419992	379992	339993	299994	259995
Real frequency (Hz)	220000	180000	140000	100000	60000	20000
Measured events (cps)	219996	179996	139997	99998	59999	20000

and increasing (or decreasing) V_T slightly.

Since the circuit was successfully tested with 0.1 V amplitude signals, it will work with no problem with weak signals from photon detectors and laser sync out signals (section 2.2).

5.2. Event frequency measurement

This section describes test results for the application introduced in section 3.4.1. TTL signals of amplitude 1.8 V and different frequencies were generated by using a digital delay/pulse generator (model DG535 from Stanford Research Systems). It can create signals with frequencies as high as 1 MHz. This signal was connected to port IO2(0) in the FPGA.

Measurements for frequencies between 20 kHz and 700 kHz were taken. Results are shown in table 5.1.

5.2.1. Analysis

In every case the percent error is under 0.0025%. The relative error is presented in figure 5.4. This error can even be attributed to the signal generator itself, since the absolute

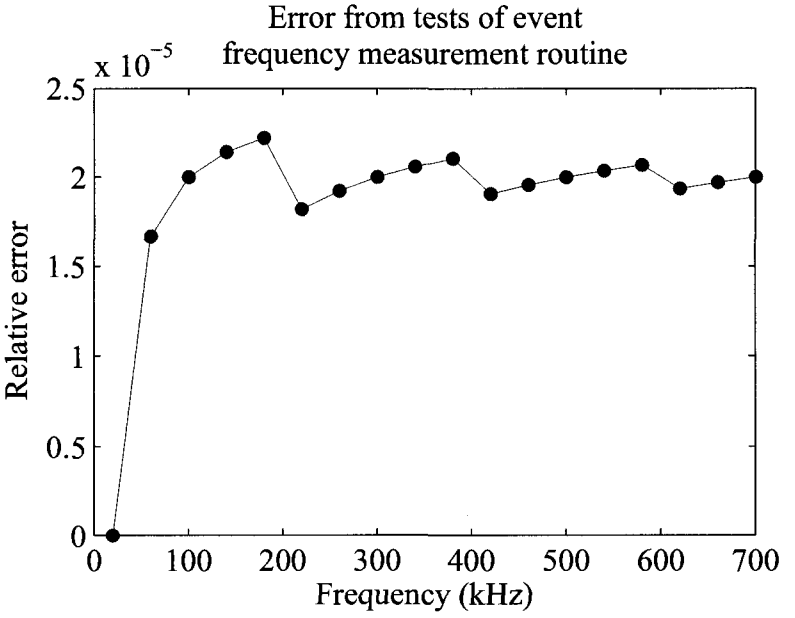


Figure 5.4 Absolute error obtained on event frequency measurement application tests. For low frequencies, error is 0. For high frequencies the error oscillates around a limit, about $2 \times 10^{-5} = 0.002\%$.

error shows regions where it increases, followed by a sudden decrease, instead of a monotonous increase with the signal frequency. For low frequencies $f < 20$ kHz the error is zero.

In conclusion, we have an accurate count per second value as output of the designed system with the FPGA.

5.3. Coincidence detection

This application described in section 3.4.2 was tested using a digital delay/pulse generator (model DG535 from Stanford research systems), which is capable of generating two signals in two different channels A and B, where the signal in B is delayed a controllable finite time with respect to the signal in A. The finest resolution of this delay is 5 ps. Signals of 500 kHz are used for these tests.

Identical cables were used to connect the signals A and B from the delay generator to the FPGA input ports IO2(0) and IO2(1) respectively. A LabView interface (figure 5.5) is made, allowing to check coincidences between channels A and B as well as the counts per second in each channel (event frequency measurement routine).

The interface was tested in a quantum optical experiment, where correlated photons at two different wavelengths are produced by a non linear crystal. If the optical line is well aligned, coincidences should be detected at the end of the line. Therefore, this application serve to align the optical set-up, by checking for coincidences and counts per second per channel.

5.3.1. Synchronous solution

The synchronous approach was implemented with $T_{\text{CLK}} = 5$ ns, and a waiting time $t_{\text{wait}} = 2T_{\text{CLK}} = 10$ ns.

The expected result is to have a coincidence when signals A and B differ by less than 10 ns. When the signals differ by 15 ns or more, the system should not count this events as coincidences. In the time window $10 \text{ ns} < t < 15$ ns the system counts it as coincidence with a certain probability. This happens since event's arrivals between $m \cdot T_{\text{CLK}}$ and $(m+1) \cdot T_{\text{CLK}}$ cannot be distinguished, where $m \in \mathbb{N}$ (see figure 5.6).

Changing the delay from 0 ns to 20 ns, and taking several measures per delay, the result shown in figure 5.7 is obtained. The coincidences are normalized, dividing the coincidences per second by the counts per second of the channels (always the same for both channels).

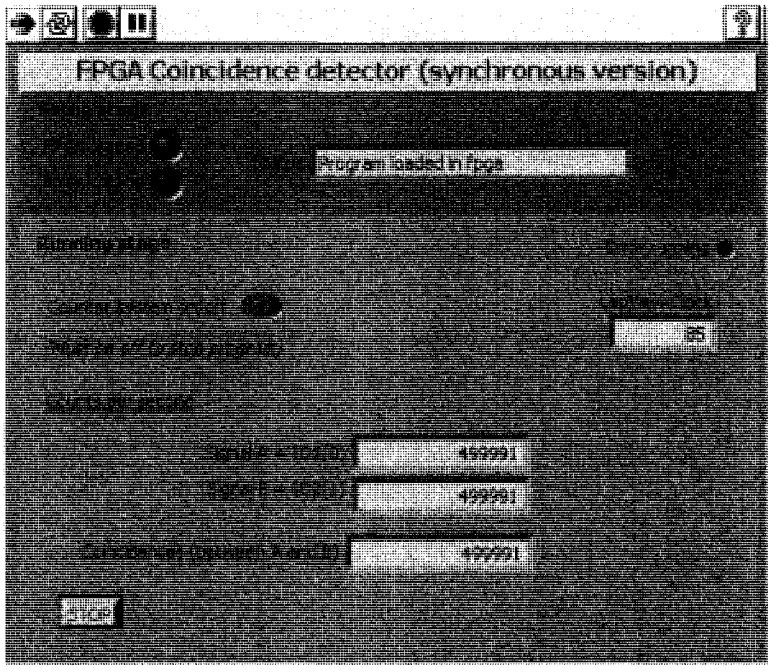


Figure 5.5 Front panel of coincidence detection application in LabView.

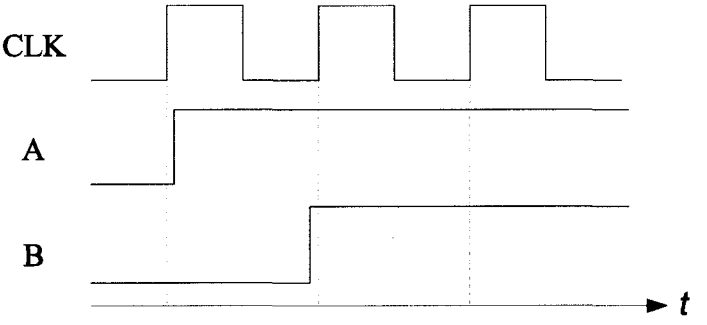


Figure 5.6 Synchronous systems cannot distinguish between signals that arrive within the same T_{CLK}. In this example A and B are equal for the system, since it reads them each clock's rising edge.

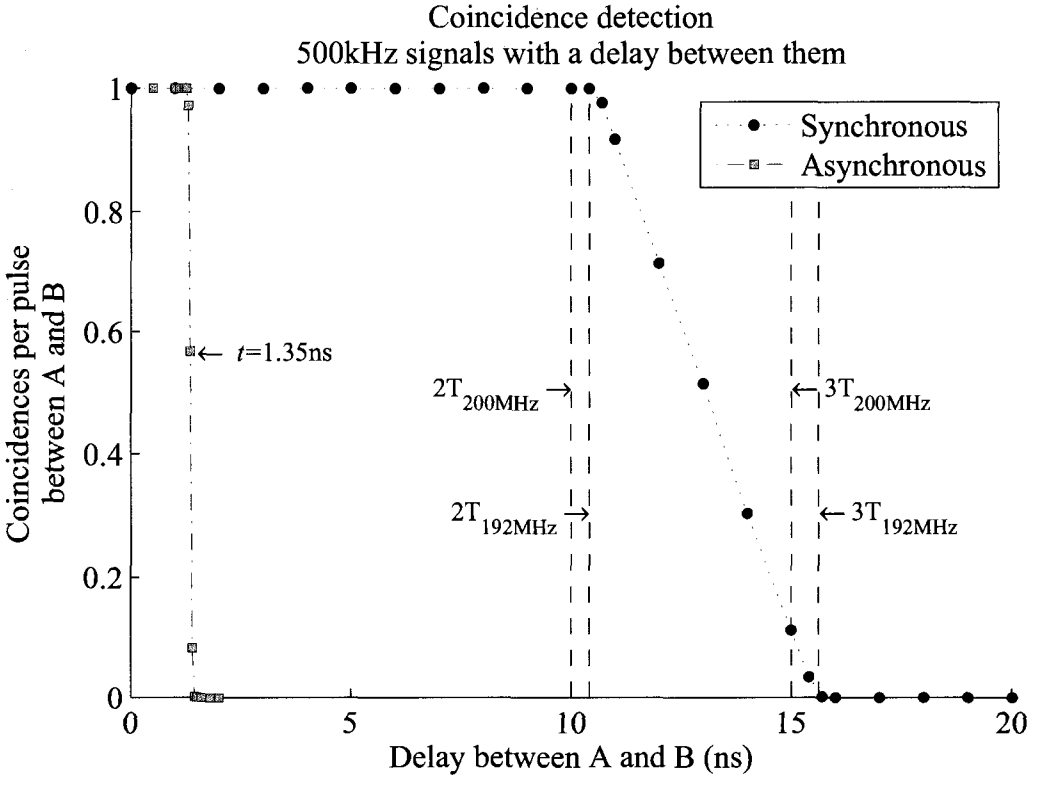


Figure 5.7 Coincidence detection results. By changing the delay between two 500 kHz TTL signals, normalized coincident events are here shown. Synchronous approach: for $t_{\text{delay}} < 2T$ events count as coincidences; for $t_{\text{delay}} > 3T$, events count as non-coincidences, T being the system's clock period. Asynchronous approach: for $t_{\text{delay}} < 1.25$ ns, events count as coincidences; for $t_{\text{delay}} > 1.5$ ns, events count as non-coincidences.

5.3.2. Asynchoronous solution

Tests of coincidence detection with the asynchronous approach were made as well. The expected result is to have coincidences for signals with almost no delay between them, since the permitted delay corresponds to the execution time of two consecutive NOT-gates. In contrast to the synchronous solution, a fast transition zone is expected in terms of delay.

A sweep of delay times from 0 ns to 2 ns is done. The results are in figure 5.7. Signals with less than a 1.35 ns delay are considered as coincidences; when the delay is greater, signals are seen as non coincident. A transition happens between 1.25 and 1.5 ns.

5.3.3. Analysis

The overall behaviour is the expected one: for a reduced time delay, all the events are counted as coincidences; for a high time delay, no coincidences are detected; for a specific delay window, the probability of taking events as coincidences ranges between 0 and 1. The moments where these three behaviours should occur for the synchoronous approach are at $2T_{\text{CLK}}$ and $3T_{\text{CLK}}$. The results (figure 5.7) show that this does not happen at 10 ns and 15 ns as expected; however is not far from those values. If the times are marked for a clock frequency $f_{\text{CLK}} = 192$ MHz, the window where the probabilistic behaviour takes place, fits with the predicted situation.

For the asynchronous version, the time that the system waits for considering two signals as a coincidence is considerably shorter than the fastest synchronous detection system (15 times shorter). If high accuracy is required, the asynchronous version is the best choice. The asynchronous system presents an unexpected feature: the transition zone. Despite of this non-instantaneous change between coincidences and no-coincidences

(around 1.35ns, figure 5.7), the transition is much faster than a T_{CLK}.

In order to have a synchronous coincidence detection system with similar characteristics as its asynchronous counterpart, it would be necessary to have a clock of

$$T_{\text{CLK}} = (1.5 - 1.25)\ \text{ns} = 250\ \text{ps} \tag{5.1}$$

or in frequency terms, a $f = 4$ GHz clock.

A conclusion from the taken measurements is that the generated frequency of the DCM is not exactly 200 MHz. More importantly, both synchronous and asynchronous coincidence detection routines reliably distinguish between coincidence and non-coincidence events.

5.4. Time stamping

An application which records 5 input signals was tested. The description of its working principle is presented in section 3.4.3.

Data was obtained from two silicon detectors (EG&G silicon single photon counting modules). They provide a 1.8 V TTL pulse of about 150 μs per click, with a dead time of about 250 μs. They saturate at about 1 million counts per second.

The optical setup consisted of a system that was designed to provide entangled photon pairs at 760 nm and 680 nm. These optical signals were optically separated (by a demultiplexor) and then measured with the above named silicon detectors. These signals were used as inputs in ports IO2(2) (channel 2) and IO2(3) (channel 3) respectively at the FPGA. The counts (obtained using an SR400 two channel gated photon counter from Stanford Research Systems) of channel 2 were about 55000 c.p.s. (counts per second),

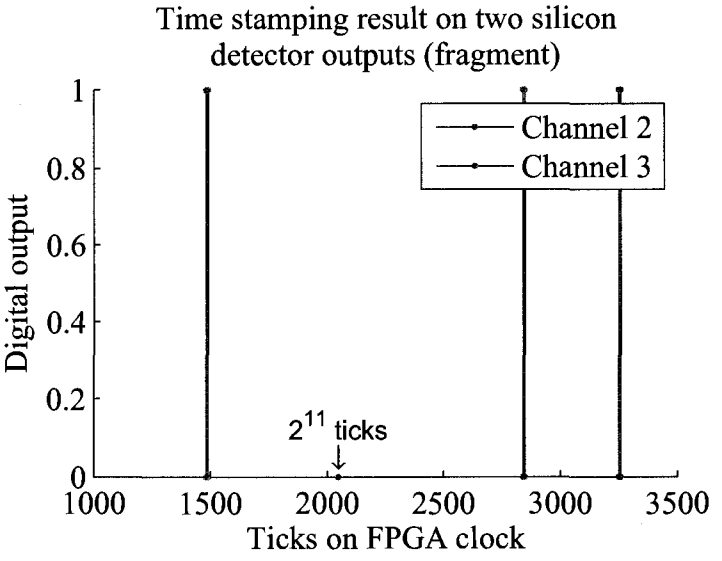


Figure 5.8 Time stamping. Reconstruction of two single photon detectors signals. Data is recorded when a change in any of the tracked signals occurs (points). Also each 2^{11} clock ticks data is recorded, for whole signal reconstruction. The presented fragment ocurred at $29 \cdot 2^{11}$ ticks away from tick zero. Sampling frequency = 48 MHz.

and for channel 3 about 8800 c.p.s.

Since the bus of transferred data between the computer and the FPGA is of 16 bit (see section 3.2), the bus is composed by the digital binary version of the 5 input pins, and the other 11 bits are used for the time stamp. This means that if there is no change at any input signal, the FPGA sends data each $2^{11} = 2048$ clock cycles. At 48 MHz,

$$t_{\text{wait}} = \frac{2^{11}}{48 \times 10^6 \text{ Hz}} = 42.66 \ \mu\text{s}. \tag{5.2}$$

A plain text file with this information is obtained.

5.4.1. Data processing and analysis

A sample of the taken data is shown in figure 5.8. This is the reconstruction of a fraction of the signal, after processing the data. Each time that a signal (channel 2 or channel 3) shows a change, the state of the 5 signals is received with its corresponding time (in clock ticks). Each 2^{11} ticks, with or without modification of the recorded signals, the state is recorded, and the internal FPGA counter starts over.

The duration of the measured pulses was 7 or 8 ticks; this corresponds to 145.83 and 166.6 ns. This means that the pulse duration is about 150 ns.

The pulses per second at both detectors were calculated. At channel 2, 46638 pulses were detected per second; 15.2% of error compared with the previously counted pulses. At channel 3, 5414 pulses were measured; 38.5% of error in this case. In the disconnected channels no pulse was registered.

The missed events are mainly lost information in the recording process. This could be noticed since in a file of about 125000 registers, 94 times the data that corresponds to 2^{11} ticks is missing. This irregularity can be found by verifying how many times the clock decreases. Overflowing any memory involved in the process (FPGA, USB interface, computer) can cause this situation.

A 100 kHz signal from a delay/pulse generator was used to test the time stamping application once again. 99957 pulses per second were obtained; 0.043% error is present. However, the missing control ticks were 84 of 512000 registers.

Also the coincidences can be determined between any two channels by processing the recorded information; about 24.5 coincidences per second were detected between channels 2 and 3. Calculated coincidences show that only dark-count type coincidences are detected. The recorded pulses are close in the corresponding orders of magnitude for

Table 5.2 Periodic signal generator. Measured output characteristics.

Characteristic	**45 MHz**	**45 MHz with π phase**	**90 MHz**
Frequency	45.004 MHz	45.147 MHz	96.246 MHz
Period	22.22 ns	22.15 ns	10.39 ns
Rise time	930 ps	900 ps	1.07 ns
Fall time	850 ps	830 ps	960 ps
Duty cycle	49.5%	50.9%	53.0%

each detector signal. However a considerable loss of pulses is noticed.

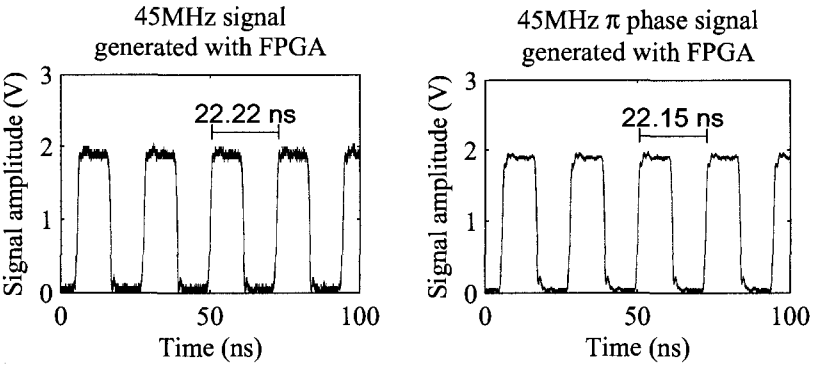
5.5. Periodic signal generator

Periodic signals are obtained from the FPGA by using the scheme described in section 3.4.5. In particular 45 MHz and 90 MHz signals are generated. The characteristics of the output signals were measured with a 500 MHz oscilloscope (Yokogawa DL1700 model); they are shown in table 5.2. They were taken from port IO2(0).

The resulting waveforms are shown in figure 5.9. These three signals are obtained from the same DCM (Digital Clock Manager) with a clock input of $f_{\text{CLKIN}} = 48$ MHz, and were measured at the same output pin. The outputs of the DCM that were taken correspond to CLKFX for the main output (45 MHz) CLKFX180 for the main output with a π phase, and CLK2X with the double of the frequency (90 MHz).

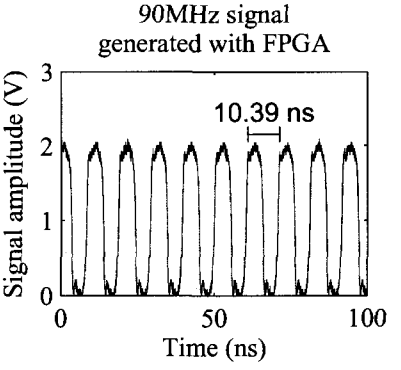
5.5.1. Analysis

Since the main synthesized frequency is 45 MHz ($T = 22.2$ ns), its waveform has the best performance in terms of frequency and duty cycle (50% is desired). Note that the signal that is supposed to have 90 MHz ($T = 11.1$ ns) has actually about 96 MHz ($T = 10.4$ ns). To generate a more accurate signal in frequency, it is suggested to work



(a) 45 MHz.

(b) 45 MHz with π phase.



(c) 90 MHz.

Figure 5.9 Periodic signal generator output waveforms for the three programed values.

with a DCM whose main frequency is set at 90 MHz. Is good to remember that the DCM resources are very limited; four in the XC3S2000 model and two are required by the ZestSC2 card.

5.6. Output circuit

An amplitude modulator channel at the output circuit was tested. This circuit is described in section 4.3. Refer to figure 4.3 to see the schematic of the circuit.

Care has to be taken with ground signals. Supply voltages of the output circuit and the ZestSC2 card should come from sources not coupled to ground. The USB port may also provide an undesired ground coupling.

The measured values of the used resistances are 71.37 Ω and 3330 Ω, which placed in parallel give a resistance of

$$R_A = \left(\frac{1}{71.37\ \Omega} + \frac{1}{3300\ \Omega}\right)^{-1} = 69.87\ \Omega \tag{5.3}$$

If the resistance of the load (i.e. the amplitude modulator) is 50 Ω, when the transistor conducts the voltage at the load (modulator) is

$$V_A = (V_{CC} - V_{EE})\frac{50\ \Omega}{R_A + 50\ \Omega} \tag{5.4}$$

$$= 12\ \text{V}\frac{50\ \Omega}{119.87\ \Omega} \tag{5.5}$$

$$= 5.0053\ \text{V} \tag{5.6}$$

and when the transistor does not conduct, $V_A = 0$ since no current pass through the output SMA connector. In practice, the voltage V_{GS} of the transistor is $-V_{EE} \neq 0$ V, and therefore the current at the drain i_D is not zero. As consequence, the output SMA

connector voltage is not zero.

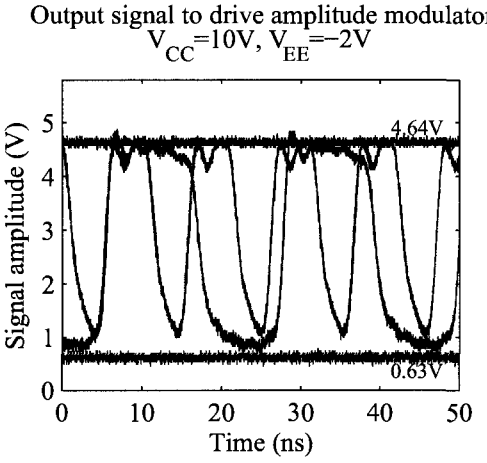
The periodic signals of 45 MHz and 96 MHz obtained at section 5.5 were used for testing purposes. Their 2 V amplitude difference at the input of the circuit suffices to turn the transistor on (off) and provide the current needed to obtain a signal with the desired amplitude.

Tests were made by changing V_{CC} and V_{EE}. The output voltage at the SMA connector was about 4.67 V, by maintaining a difference of $V_{CC} - V_{EE} = 12$ V and provided a DC coupling at 50 Ω (equivalent to have connected an amplitude modulator). The operation supply voltages are $V_{CC} = 10$ V and $V_{EE} = -2$ V (see section 4.3.1); measured results are shown in figure 5.10(a). The voltage for a logic '1' is 4.64 V, where the current through the channel is close to the required limit for correct functioning. Reducing V_{GS} would reduce i_D, and therefore the output voltage would not be the result of a voltage divider but the maximum voltage that can be obtained with the particular i_D current.

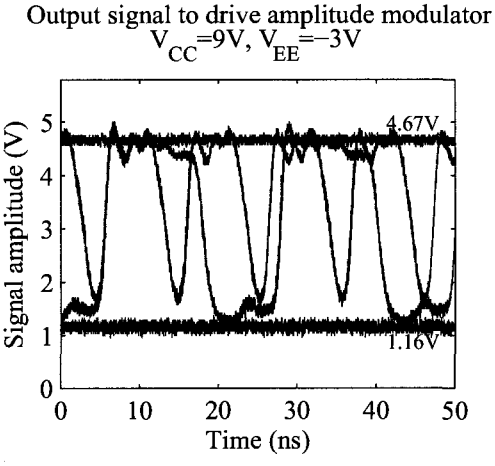
A second case was tested, where the $(V_{CC} - V_{EE}) = 12$ V condition was respected, but the supply voltages were changed to work closer to the threshold voltage of the transistor (about 3 V), such that $V_{GS} = V_{th}$. This situation, whose measurements are depicted in figure 5.10(b), shows a better response in terms of i_D for a logic '1' input. However to work that close to V_{th} results in increasing the output voltage for a '0' logic input; the ideal case would be a 0 V output for a '0' input.

An output of 5 V was obtained, with $V_{CC} = 10.14$ V and $V_{EE} = -3.04$ V; this corresponds to inhibit light from passing trough the electro-optical modulator, given that $V_\pi = 5$ V. However, by operating at these voltages the output at zero level increases, and reaches values that are higher than 1 V. This result is shown in figure 5.10(c).

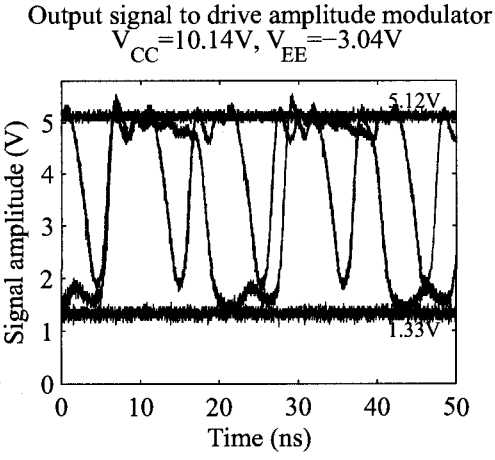
The responses at 45 MHz and 90 MHz have a fast rising edge response, but the falling edge response is not fast enough. The voltage level decreases exponentially with the



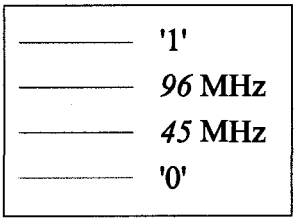
(a) Test with $V_{CC} = 10$ V and $V_{EE} = -2$ V. Critical i_D obtained at logic '1' condition.



(b) Test with $V_{CC} = 9$ V and $V_{EE} = -3$ V. Undisired i_D rise easier at logic '0'.



(c) Test to obtain an output of 5 V for a logic '1' input, resulting in $V_{CC} \approx 10$ V and $V_{EE} \approx -3$ V.



(d) Legend for figures 5.10(a), 5.10(b) and 5.10(c).

Figure 5.10 Results from tests on custom built output circuit, at a single amplitude modulator output. Voltages of 5 V and 0 V are expected. Frequency response is also tested.

same time constant, despite of the frequency variation; hence this is doubt to a capacitive load. This has as consequence that increasing the input frequency results in a '0' level that does not arrive to the supposed value. Notice that this characteristic is not heralded of the FPGA output, as can be verified in figure 5.9. A possible improvement to this situation is to take into account capacitive loads in the PCB design, or adding inductive loads to the circuit.

CONCLUSION

An electronic processing system for quantum communication and computing experiments was developed. Electronic systems are useful and at the same time unavoidable in these types of experimental realizations. The developed system offers a flexible solution that is adaptable to a wide range of situations, thanks to the signal standardization printed circuit that was designed, realized and tested in this project.

Particular applications were implemented such as event frequency measurement, coincidence detection, time stamping, pseudo-random number generation and periodic signal generation. Event frequency measurement works with outstanding performance, where the error in the measured frequency is always under 0.0025%.

Coincidence detection was done by two approaches: a synchronous and an asynchronous one. Both have been shown to reliably discern between coincident and non-coincident scenarios. For the asynchronous version, the coincidence window has a duration approximately 1.35 ns. An equivalent synchronous version would require a clock of the order of 4 GHz. For the synchronous version, the coincidence window depends on the system's clock period; at $T_{\text{CLK}} = 200$ MHz this time ranges between 10 ns and 15 ns.

Time stamping application provides a way to reconstruct up to 5 recorded digital signals. The number of recorded signals can be modified, having as consequence a change in the recording overflow time. Timing information is useful for post-processing analysis such as coincidence detection, signal correlation and arrival statistics. This is relevant to quantum communication experiments, where the count rate can be about 10^2 to 10^4 counts per second.

To control phase and amplitude modulators, periodic signals were generated at 45 MHz and 96 MHz. Their amplitude is corrected and amplified (or reduced) using the printed

circuit designed and made for this purpose. This circuit was also tested, where changes in supply voltages and resistance values suffice to obtain a determined phase or amplitude modulation. For a designed operation of 0 V to 5 V, the obtained output voltages are 0.63 V and 4.67 V respectively. Frequency response was tested successfully.

Pseudo-random series of numbers were done by using an LFSR Galois. This could be used to play the role of a coin in quantum communication protocols that require a random number selection in their process.

The system shows several advantages. Among the most remarkable are its flexibility given by its reprogrammable capacity, the adaptability to other domains, a centralized management of information, and an easy project expansion. Also, it is possible to offer graphical interfaces of simple use, like those developed in LabView.

The codes to program the FPGA are particularly adapted to the ZestSC2 card. Thanks to the used block design, they are easily adaptable to other types of cards, or even to FPGAs from other producers.

The main limitation of the present work had been the processing speed. By default the system clock runs at 48MHz. Changing it to other frequencies requires extra effort. Furthermore, the available frequencies are limited by the FPGA's characteristics; the used device in particular does not reach higher operating speeds than 210MHz.

Using it in fibre optic systems with time-bin encoding, the speed limit translates to an interferometer size constraint. For example a 200 MHz (5 ns) clock, translates into a 1.5 m delay at the interferometer.

The continuity of this project is highly recommended. New applications and functions can be integrated, making more profitable the utilisation of the final product. It can even be exploited in other research areas.

A researcher who wishes to continue the project may take all the applications here presented, execute them, study and understand the corresponding source codes in each language (VHDL, C, LabView). As a first task, I suggest improving the coincidence detection utility by adjusting FPGA's inner delay times.

REFERENCES

AN1294 (2001). *AN1294 Application Note. PowerSO-10RF: the first true RF power SMD package.* STMicroelectronics.

Bennett, C. H. and Brassard, G. (1984). Quantum cryptography: public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India*, pages 175–179.

Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., and Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, **70**(13), 1895–1899.

Berlín, G., Brassard, G., Bussières, F., and Godbout, N. (2008a). A fair loss-tolerant quantum coin flipping. *Quantum Communication, Measurement and Computing (QCMC), 2008 Proceedings of The Ninth International Conference on.*

Berlín, G., Brassard, G., Bussières, F., and Godbout, N. (2008b). Loss-tolerant quantum coin flipping. *Quantum, Nano and Micro Technologies, 2008 Second International Conference on*, pages 1–9.

Brendel, J., Gisin, N., Tittel, W., and Zbinden, H. (1999). Pulsed energy-time entangled twin-photon source for quantum communication. *Phys. Rev. Lett.*, **82**(12), 2594–2597.

Brooks, D. (1998). Pcb impedance control: Formulas and resources. *Printed Circuit Design Magazine.*

Clark, D. W. and Weng, L.-J. (1994). Maximal and near-maximal shift register sequences: Efficient event counters and easy discrete logarithms. *IEEE Transactions on Computers*, **43**, 560–567.

DS099 (2008). *Spartan-3 FPGA Family Data Sheet.* Xilinx, Inc.

Engel, A., Semenov, A., Hübers, H., Il'in, K., and Siegel, M. (2004). Superconducting single-photon detector for the visible and infrared spectral range. *Journal of Modern Optics*, **51**(9), 1459.

Gisin, N., Ribordy, G., Tittel, W., and Zbinden, H. (2002). Quantum cryptography. *Reviews of Modern Physics*, **74**(1), 145+.

Karlsson, A., Bourennane, M., Ribordy, G., Zbinden, H., Brendel, J., Rarity, J., and Tapster, P. (1999). A single-photon counter for long-haul telecom. *Circuits and Devices Magazine, IEEE*, **15**(6), 34–40.

Kitaev, A. (2002). Quantum coin flipping. MSRI lecture, http://www.msri.org/publications/ln/msri/2002/qip/ kitaev/1/index.html.

Komiyama, S., Astafiev, O., Antonov, V., Kutsuwa, T., and Hirai, H. (2000). A single-photon detector in the far-infrared range. *Nature*, **403**(405).

LT1226 (1992). *LT1226 Low Noise Very High Speed Operational Amplifier datasheet.* Linear technology corporation.

lvexcode (2003). *Using external code in LabVIEW.* National Instruments Corporation.

Nielsen, M. A. and Chuang, I. L. (2000). *Quantum computation and quantum information.* Cambridge University Press, New York, NY, USA.

Raussendorf, R. and Briegel, H. J. (2001). A one-way quantum computer. *Phys. Rev. Lett.*, **86**(22), 5188–5191.

Rosfjord, K., Yang, J., Dauler, E., Kerman, A., Anant, V., Voronov, B.M., G. G., and Berggren, K. (2006). Nanowire single-photon detector with an integrated optical cavity and anti-reflection coating. *Optics Express*, **14**(2), 527.

Saleh, B. and Teich, M. (1991). *Fundamentals of Photonics.* John Wiley & Sons, New York.

Shannon, C. (1949). Communication in the presence of noise. *Proceedings of the IRE*, **37**(1), 10–21.

Sweeney, C. and Bowen, M. (2006). *ZestSC2 User Guide.* Orange Tree Technologies Ltd.

UG331 (2008). *Spartan-3 Generation FPGA User Guide.* Xilinx, Inc.

Wootters, W. K. and Zurek, W. H. (1982). A single quantum cannot be cloned. *Nature*, **299**, 802–803.

APPENDIX I

PCB PIN CONNECTIONS

I.1 Input circuit connections

The circuit has 8 connectors as shown in figure I.1: 6 JTAG headers with the J prefix (Jx), and 2 of them with the SV prefix (SVx). The J1 connector is intended for the main supply voltage, 3.3V. The J2, J3, J4 nd J5 connectors provide access to the eight V_{ref} and V_T. Notice that there are two ways of providing V_{ref} voltages: if the LM4880 and the potentiometer are placed, this voltage is done by the voltage divider; if this components are not placed these voltages can be provided externally using the J2 and J3 connectors.

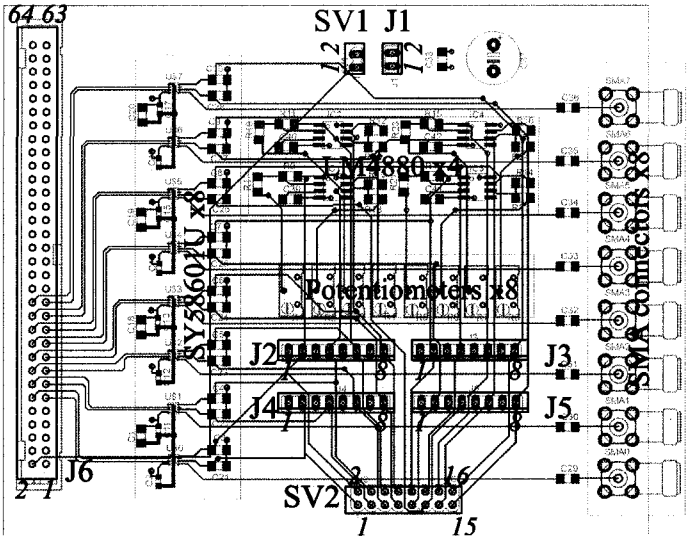


Figure I.1 Layout of PCB for signal conversion.

J6 connector is designed to fit in the J8 connector of the ZestSC2 FPGA USB card. The pins where a differential pair Q, /Q arrives (from pin 11 to pin 26) were chosen in such a way that they correspond in the FPGA also to differential pair pins.

Table I.1 Pins for J1 in input PCB: JTAG header for voltage supply

Pin	Signal
1	GND
2	V_{CCO}

Table I.2 Pins for J2 in input PCB: JTAG header for $V_{\text{ref}}(0..3)$

Pin	Signal
1	GND
2	VREF0
3	GND
4	VREF1
5	GND
6	VREF2
7	GND
8	VREF3

The SVx connectors are designed to shunt them optionally. SV1 will short circuit the V_{CCO} from the PCB with the $V_{\text{FPGA-CCO}}$ from the FPGA. SV2 provides a way to connect each $V_{\text{ref}}(i)$ with its corresponding $V_T(i)$.

Table I.3 Pins for J3 in input PCB: JTAG header for $V_{\text{ref}}(4..7)$

Pin	Signal
1	GND
2	VREF4
3	GND
4	VREF5
5	GND
6	VREF6
7	GND
8	VREF7

Table I.4 Pins for J4 in input PCB: JTAG header for $V_T(0..3)$

Pin	Signal
1	GND
2	VT0
3	GND
4	VT1
5	GND
6	VT2
7	GND
8	VT3

Table I.5 Pins for J5 in input PCB: JTAG header for $V_T(4..7)$

Pin	Signal
1	GND
2	VT4
3	GND
4	VT5
5	GND
6	VT6
7	GND
8	VT7

Table I.6 Pins for J6 in input PCB: JTAG header to be connected with FPGA (J8 of ZestSC2 card).

Signal	Pin	Pin	Signal
VCC (5V from FPGA)	1	2	$V_{\text{FPGA-CCO}}$
GND	3	4	GND
GND	5	6	GND
GND	7	8	GND
NC	9	10	NC
Q0	11	12	/Q0
Q1	13	14	/Q1
Q2	15	16	/Q2
Q3	17	18	/Q3
Q4	19	20	/Q4
Q5	21	22	/Q5
Q6	23	24	/Q6
Q7	25	26	/Q7
NC	27	28	NC
NC	29	...	NC
NC	...	56	NC
NC	57	58	GND
GND	59	60	GND
GND	61	62	GND
GND	63	64	GND

Table I.7 Pins for SV1 in input PCB: JTAG header to shunt supply voltages

Signal	Pin	Pin	Signal
$V_{\text{FPGA-CCO}}$	1	2	V_{CCO}

Table I.8 Pins for SV2 in input PCB: JTAG header to shunt reference voltages

Signal	Pin	Pin	Signal
VT0	1	2	VREF0
VT1	3	4	VREF1
VT2	5	6	VREF2
VT3	7	8	VREF3
VT4	9	10	VREF4
VT5	11	12	VREF5
VT6	13	14	VREF6
VT7	15	16	VREF7

Table I.9 SMA connectors in input PCB

Connector	Signal
SMA[0..7]	$V_{\text{in}}[0..7]$

I.2 Output circuit connections

The circuit for the electro-optic modulators has 2 connectors as shown in figure I.2: a JTAG header J1 of 3 pins, and X3 which is a 32x2 pin header connector.

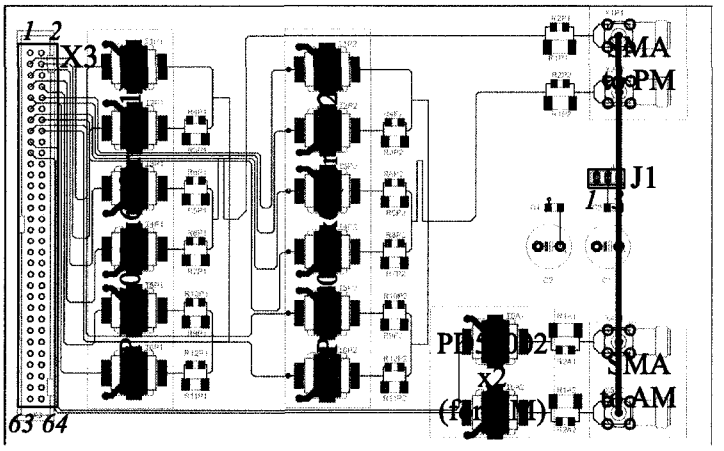


Figure I.2 Layout of PCB for signal conversion.

X3 connector is designed to fit in the J9 or J10 connectors of the ZestSC2 FPGA USB card. If it is connected to J9, the signals from the FPGA will be IO2(x); if it is connected to J10, the signals from the FPGA will be IO7(x). For phase modulators, signals are TxPy, where Tx indicates the number of the transistor ($x = \{1, 2, 3, 4, 5, 6\}$) and Py the number of the phase modulator ($y = \{1, 2\}$). For amplitude modulators signals are T0Az since there is only one transistor, and Az indicates the number of the amplitude modulator ($z = \{1, 2\}$).

Table I.10 Pins for J1 in output PCB: JTAG header for voltage supply

Pin	Signal
1	V_{EE}
2	GND
3	V_{CC}

Table I.11 Pins for X3 in output PCB: JTAG header to be connected with FPGA (J9 or J10 of ZestSC2 card).

Signal	Pin	Pin	Signal
NC	1	2	NC
T1P1	3	4	T2P1
T3P1	5	6	GND
T4P1	7	8	T5P1
T6P1	9	10	T1P2
T2P2	11	12	GND
T3P2	13	14	T4P2
T5P2	15	16	T6P2
T0A1	17	18	GND
T0A2	19	20	NC
NC	21	22	NC
NC	23	24	GND
NC	25	26	NC
NC	27	28	NC
NC	29	30	GND
NC	31	32	NC
NC	33	34	NC
NC	35	36	GND
NC	37	38	NC
NC	39	40	NC
NC	41	42	GND
NC	43	44	NC
NC	45	46	NC
NC	47	48	GND
NC	49	50	NC
NC	51	52	NC
NC	53	54	GND
NC	55	56	NC
NC	57	58	NC
NC	59	60	GND
NC	61	62	NC
NC	63	64	GND

Table I.12 SMA connectors in output PCB

Connector	**Signal**
X1P1	Phase modulator 1
X2P2	Phase modulator 2
X4A1	Amplitude modulator 1
X5A2	Amplitude modulator 2